UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DEGRAU

Grau en Enginyeria Electrònica i Automatització Industrial

SISTEMA ROBÒTIC DE CLASSIFICACIÓ D'OBJECTES EN PERSPECTIVA AMB VISIÓ PER COMPUTADOR

Memòria

Autor: Eudald Ballescà Casas
Director: Antoni Grau Saldes

Convocatòria: Maig 2018

Resum

Aquest document descriu el disseny del procés que permet el reconeixement d'objectes geomètrics senzills des de qualsevol angle i amb el major ventall d'inclinacions possibles. Amb aquest objectiu i després de realitzar una cerca dels sistemes actuals que permeten la identificació de la posició relativa entre dos objectes, transformació de la imatge i identificació d'objectes que hi apareixen, s'ha optat per desenvolupar un sistema simplificat de reconeixement de posició basat en els sistemes de realitat augmentada.La transformació d'imatges aplicant la relació de la projecció cònica frontal a pla zenital del dibuix clàssic i el reconeixement d'objectes mitjançant la identificació de propietats de una figura com area, perímetre, relació entre radi màxim i mínim, circularitat o numero de vèrtex.

S'utilitzarà una càmera web convencional amb la mínima resolució possibleper a que el codi s'executi ràpidamentperò garantint resultats òptims. Per a l'execució del programa serà necessaria l'extensió d'adquisició d'imatges de MATLAB, tot i que el programa esta realitzat amb el mínim de funcions predefinides possibles per així facilitar la conversió del mètode utilitzat a altres llenguatges.

Els resultats permeten enviar satisfactòriament la informació de la posició a la realitat necessària per seleccionar quin objecte es vol.

Resumen

Este documento describe el diseño del proceso que permite el reconocimiento de objetos geométricos sencillos desde cualquier ángulo y en el mayor abanico de inclinaciones posibles. Con este objetivo y después de realizar una búsqueda de los sistemas actuales que permiten la identificación deposicionamiento relativo entre dos objetos, transformación de imagen e identificación de los objetos que se encuentran en esta, se ha optado por desarrollar un sistema simplificado de reconocimiento deposición basado en los sistemas de realidad aumentada, la transformación de imágenes aplicando la relación de la proyección cónica frontal a plano cenital del dibujo clásico y el reconocimiento de objetos mediante la identificación de propiedades de una figura como el área, perímetro, relación entre radio máximo y mínimo, circularidad o el numero de vértices.

Se utilizara una camera web convencional con la mínima resolución posible para hacer que el código se ejecute rápidamente pero garantizando que se obtengan los resultados óptimos, para la ejecución del código será necesaria la extensión de adquisición de imágenes de MATLAB, aunque todo el código se haya realizado usando el mínimo de funciones predefinidas de MATLAB posibles para facilitar la conversión de este método a otros lenguajes.

Los resultados permiten mandar satisfactoriamente la información necesaria para encontrar la posición en la realidad del objeto que se seleccione.

Abstract

This document describes the design of the process that allows the recognition of simple geometrical objects from any angle and the most inclinations possible. With this objective and after researching the actual systems that allow the identification of relative position between two objects and image transformation and recognition of the previously mentioned, a simplified position recognition system has been developed based in augmented reality, the image transformation applying the frontal conic projection relation to a zenithal plan of traditional drawing and the object recognition using the property identification of a figure such as area, perimeter, relation between maximum and minimum radius, circularity or the number of vertices.

A conventional webcam is used with the lowest resolution possible to execute the code in short order but ensuring optimal results.

To execute the code MATLAB'S image acquisition extension will be needed even if the coding has been done using minimum predefined functions to provide an easy conversion from this method to another languages.

As a result, it is possible to send the necessary information to find the position in reality of the selected object.

Glossari

TFG: Treball de final de grau

TAG: Derivat del terme ARTag que fa referència a les marques per a dispositius de realitat augmentada necessàries per ajudar a determinar a quina distancia es troba un punt de la càmera i en quina orientació en temps real per poder reproduir objectes virtuals en una pantalla. En aquest cas un sistema més simple.

Variables

En aquest apartat apareixen les variables més comunes, totes les variables utilitzades s'expliquen en el inici del pseudocodi on apareixen.

Y_Fuga: Valor de l'eix Y en el que es troba la línea d'horitzó on concorren totes les línees paral·leles.

X_Fuga: Valor de l'eix X sobre l'horitzó de Y_Fuga del que es realitzaran les projeccions que determinaran l'amplada.

Y_Base: Valor de l'eix Y en el que es troba la línea d'horitzó on es considera que la informació en punts inferiors es irrellevant.

Altura: És la altura de l'espectador respecte Y_Fuga.

X_Altura: Punt sobre l'horitzó de Y_Fuga des de on es projectaran les línees que determinaran la profunditat dels punts de la esquerra de la imatge.

X_Alturae: Com X_Altura però per als punts de la dreta de la imatge.

β: Horitzó de Y_Base, s'utilitzarà per indicar els punts que es projecten mitjançant X_Fuga com els següents:

- βa: Punt de la projecció de X_Fuga a β passant per el punt a.
- βb: Punt de la projecció de X_Fuga a β passant per el punt b.
- xβ: Punt de la projecció de X_Fuga a β passant per el punt a i per el punt b.

δ: S'utilitzarà per definir punts de la mateixa forma que s'ha fet amb β però en aquest cas projectant-los de β a X_Altura o X_Alturae.

IM[]: La estructura IM[valor] s'utilitzarà en tot el codi per anomenar a les imatge tractades. Dintre de les funcions s'utilitzarà la forma:

- IM_In per a la imatge d'entrada i amb la que la funció treballarà.
- IM_Out per a la imatge que retornarà la funció.

karea: Nom que rep la matriu de les àrees diferenciades en l'algoritme Etiquetatge 5.3

DimIM: vector amb les dimensions de la imatge principal, també apareix la variació DimIM140 que realitza el mateix però amb la IM140.

Area_filtre1: Vector que deriva de la eliminació d'alguns valors de karea.

Area_IM12: Vector que deriva de la eliminació d'espais buits de Area_filtre1.

Elements_IM12: Vector que conte el llistat de figures rellevants de la IM12.

Índex

Índex de figures

1. Introducció

1.1. Origen del treball

Tot i que aquest treball està encarat al processament d'imatges el principal ús d'aquest codi es troba el camp de la robòtica industrial.

Des de fa anys la industria està transitant cap a la quarta revolució industrial on els robots guanyen un gran pes en els processos. Aquestes maquines tant versàtils permeten la realització de moltes tasques substituint o complementant llocs de feina tediosos o de gran desgast físic.

En tractar-se d'una màquina programable sempre havia sigut necessari un entorn absolutament controlat en el que el robot hagués de realitzar les seves funcions en uns punts determinats cada cop que un sensor l'hi indicava. Després de superar moments en els que degut a la falta de potencia en els processadors la robòtica havia quedat estancada,actualment gaudeix de bona salut a nivell global, principalment en els països amb un sector secundari mes desenvolupat. Això ha fet que sigui necessària la creació d'entorns on els robots puguin treballar conjuntament amb persones.

Per això es necessari un conjunt d'eines que permetin al robot treballar en un entorn menys controlat on els objectes que li arriben no vinguin d'una altre màquina que li subministri tot ordenat i ben col·locat, es necessari que aquest puguin localitzar on és un objecte o quin objecte és. En aquest punt va aparèixer el control per visió.

Aquest sistema de control també te inconvenients,característiques extrínseques com la il·luminació, els colors, les ombres, els materials, la posició i orientació son característiques que poden fer impossible que aquest funcioni, per això les eines han de disposar de certa flexibilitat però també robustesa. Això implica un major cost computacional que es tradueix en més temps d'execució necessari per evitar errors.

El control per visió es basa en la interpretació de les imatges captades per una càmera, ja sigui convencional, capaç de captar ones electromagnètiques en espectre visible, o per infrarojos.

Aquest treball s'origina a partir d'una reunió amb el director del TFG en la que es van plantejar diferents temes que no s'havien plantejat en el llistat de propostes al voltant del control per visió.

La proposta elegida permet el control per visió on la orientació i posició poden variar entre execució i execució oferint la flexibilitat de que automàticament l'ordinador identificarà els nous paràmetres i realitzarà el procés sense problemes.

Aquest canvi d'orientació pot ser degut per exemple al canvi d'entorn entre la empresa que dissenya una línea de producció i la empresa final que el farà servir però també respon a la necessitat que tenen moltes empreses de que les seves plantes puguin ser reutilitzades i modificades per realitzar diferents feines dins d'un mateix dia sense un cost molt alt en el canvi d'una a una altre. Per tant tot el que siguin components mòbils son una limitació menys.

El codi està basat en la metodologia utilitzada en pintura clàssica, trigonometria i l'aprofitament dels polígons geomètrics i les seves propietats, però no continua cap treball en concret sinó que comença des de zero, tot això detallat en els capítols corresponents de la memòria.

1.2. **Objectius del treball**

L'objectiu principal d'aquest treball és el desenvolupament d'un sistema que permeti corregir la deformació que rep una superfície en ser vista des d'una posició no perpendicular a ella. Amb la posició desconeguda el sistema podrà reconèixer una sèrie de figures conegudes. Per tant es podria dividir el treball principal en diferents parts:

- Coneixement de l'entorn.
- Adquisició i tractament de baix nivell de la imatge.
- Reconeixement de la relació entre el pla de treball i la càmera.
- Reconstrucció de la imatge per tal de que quedi la superfície de treball com si hagués sigut captada en pla zenital.
- Reconeixement de les figures.
- Ubicació de les figures.
- Proves del sistema

1.3. **Abast del treball**

El sistema funciona sempre i quant es respectin les indicacions del apartat 3.2 amb un rang de:
5˚ a 60˚ respecte la vertical, des de qualsevol poció al voltant i una distancia de 40 cm a 90 cm. Aquesta distancia varia segons l'angle,identificant fins a 14 objectes diferents.

El numero d'objectes està limitat a les dades introduïdes en el programa. Aquesta llista es pot ampliar sempre i quant el nou objecte es pugui diferenciar en alguna característica. Es pot observar en aquest treball que no és necessària una gran diferencia però cal tenir en compte que la discretització que es realitza en agafar una imatge ja limita la diferenciació d'objectes similars, però en afegir transformacions la informació final pot variar més encara amb un mínim canvi d'iluminació o d'angle,

per això tots els objectes queden definits per propietats que no tenen un valor fix sinó un rang i que no poden coincidir en tots el camps amb altres peces, o seria impossible diferenciar-les.

2. Estat de l'art

2.1. Introducció

En aquest apartat es mostrarà una breu descripció sobre les diferents utilitats de la visió artificial per acabar centrant-se en mètodes de conversió d'imatges i reconeixement de figures.

2.2. Usos de la visió artificial

La visió artificial neix als anys 60 amb l'objectiu de donar a les maquines (robots o ordinadors) informació del seu entorn de la mateixa manera que la obtenim els humans. Això elimina una gran barrera per a les maquines i permet substituir feines avorrides, millorar-les o analitzar situacions que per un humà seria impossible. Alguns exemples d'això son:

- Anàlisis mèdic.
- Rastrejament de persones/cotxes per obtenir patrons de moviment.
- Sistemes de seguretat.
- Control d'incendis.
- Control de punts dèbils en estructures.
- Robòtica.
- Reconeixement de patrons (cares, formes, etc.).
- Processos industrials.

2.2.1. Visió artificial per a persones invidents

La visió artificial per a persones invidents, tot i que per l'origen de la tecnologia semblava que seria una aplicació immediata, no es desenvolupa fins ara , amb l'aparició dels telèfons intel·ligents amb càmera. [1]

En l'article citat es pot veure com algunes de les aplicacions de tipus reconeixement de patrons poden ser:

Ajudar en mobilitat, on el programa identificarà objectes de l'entorn per facilitar els desplaçaments, però al tractar-se d'un processament de molt alt nivell els processadors actuals no resulten suficientment ràpids.

Ajuda en l'orientació, un sistema GPS pot indicar la direcció que s'ha de seguir però no pot avisar dels obstacles o de si es pot creuar el pas de zebra, aquí apareix la visió artificial, identificant les rutes possibles i els perills a evitar.

Accés a informació impresa, la majoria d'informació al carrer o literatura es nomes accessible per a persones amb visió correcte.

Interacció social, part del llenguatge es no verbal, un sistema de visió artificial pot interpretar les expressions dels interlocutors i comunicar-les al usuari.

2.2.2. Visió artificial per al control de qualitat

Amb la industrialització els consumidors s'han acostumat a rebre productes amb una aparença homogènia. Això és possible gràcies als processos controlats, però també a un sistema de control de qualitat que evita que, per exemple, un envàs nou tingui defectes.

Aquest control històricament ha sigut una feina realitzada per persones, però des de fa anys aquesta selecció la realitzen maquines amb més freqüència. En aquests controls es realitzen comprovacions com geometria, color, forma o existència de rebaves.

En molts sectors industrials s'aprofita la rapidesa d'aquests sistemes de control per supervisar tots els productes, com per exemple la de l'automòbil o la de l'embalatge.

En el sector automobilístic es realitzen feines com inspecció d'estampats, mecanització, pintura i rebaves. En el de l'alimentació es realitzen tècniques com el control de fruita (veure apartat 2.2.3) la comprovació dels envasos o el control dels productes càrnics processats. En el sector de l'electrònica es revisa l'estat de les soldadures.

2.2.3. Visió artificial per al control de fruita

Un altre aplicació de la visió artificial és controlar el punt de maduresa d'un fruit, camp en el que s'ha treballat des dels anys 90, moment en el que es van desenvolupar algoritmes de classificació segons el color. Al llarg dels anys s'han anat realitzant diferents variacions, algunes utilitzant càmeres infraroges per detectar la temperatura interna o controls combinats de color i forma.

2.2.4. Visió artificial per al control de posició

Per al correcte funcionament d'algunes màquines és necessari un control del sistema de distribució, com es pot veure a la imatge Figura2.1 els paquets arriben en qualsevol orientació i el robot delta simplement els agafa i els col·loca tots en la mateixa orientació, fet que permetrà facilitar en el posterior procés l'empaquetatge.

Figura2.1. **Pickand Place de HongsenIntelligent**

2.2.5. Visio artificial per controlar la posició de la càmera

Tot i que normalment el control per visió s'utilitza per controlar el moviment de l'eina d'un robot, en alguns casos com el dels robots da Vinci el control per visió es dedica a posicionar la càmera de tal manera que sempre estigui enfocant l'eina del cirurgià.

2.3. Sistemes de control per visió

En aquest apartat es comentaran altres tipus de control per visió.

2.3.1. Control de planta amb visió perpendicular.

El sistema de control més senzill, consisteix en una càmera situada en un entorn controlat i en una posició idònia, és el sistema més utilitzat ja que aquests factors li donen molta robustesa en la indústria.

2.3.2. Control de matricules (Sistema entrenat)

Es tracta d'un sistema amb un processament d'imatge moderat en el que es reconeix la matrícula i mitjançant imatges de mostra de tots els elements possibles es realitza un reconeixement per coincidència caràcter per caràcter. Aquest sistema acostuma a portar una càmera amb visió infraroja per poder funcionar de nit. [2]

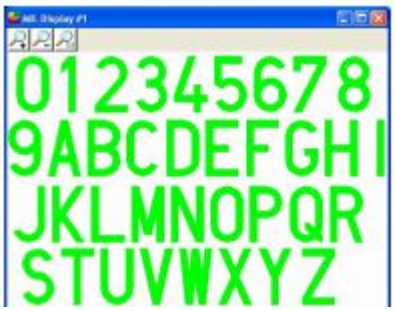

Figura2.2. **a) Plantilla conjunt caràcters** **b) Matrícula a reconèixer**[2]

2.3.3. Control de robot amb sistema estereoscòpic.

El sistema estereoscòpic permet un reconeixement 3D de l'entorn gracies a la utilització de dues càmeres.

L'adquisició d'informació es fa mitjançant la comparació de les imatges captades per cada una i coneixent la posició relativa entre elles permet poder triangular la posició dels objectes. Aquest sistema permet la obtenció de molta informació inclús en entorns no controlats però requereix molta potència de processament, i l'ús de dues càmeres no lliura el sistema de problemes com la il·luminació.[3]

2.4. Correcció de perspectiva

Actualment ja existeixen mètodes estandarditzats per a la transformació de perspectives. La més comuna és la transformació mitjançant una transformada homogènia on es projecten els punts de la imatge captada en un pla que deixi aquesta amb la forma desitjada (Figura2.3)

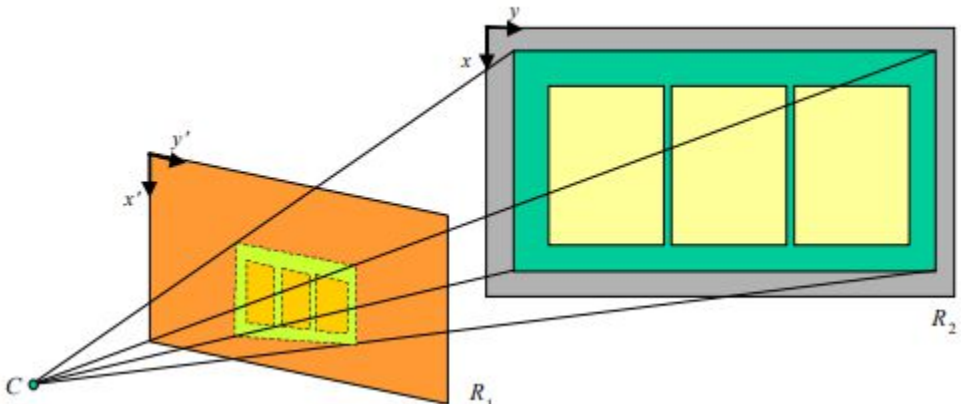

Figura2.3. **Projecció entre dos plans no paral·lels[4]**

Tot i que aquesta és la més establerta en la computació també existeixen altres tècniques prèvies a aquesta, com per exemple la que s'utilitza en dibuix on la idea és la mateixa però es realitza mitjançant la geometria on el punt de fuga pren un gran protagonisme (Figura2.4). Aquest mètode s'explica millor en l'apartat 4

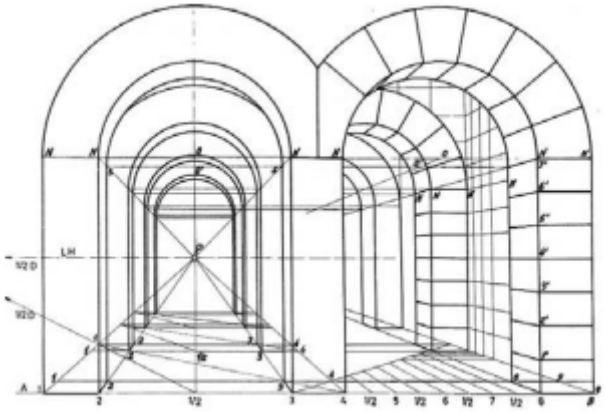

Figura2.4. **Projecció entre dos plans no paral·lels [5]**

3. Procés

En aquest capítol s'explicarà el procés que es segueix per fer servir el codi i quin procés segueix aquest.

3.1. Preparació de la planta

La planta per la que s'ha ideat el sistema consta d'un braç robòtic Festo Mitsubishi RV-2AJ protegit per 3 costats per un metacrilat transparent però obert per la part davantera, cantó per el qual es treballarà, i serà aquí on es col·locarà una base de fusta per millorar la zona de treball útil pintada de blanc (tot i que no es busca optimitzar-la ja que és innecessari), ja que això millora la detecció de tot el programa.

Sobre aquesta plataforma s'instal·laran les quatre marques (a partir d'ara TAG, 3 cercles i 1 quadrat) que s'utilitzaran com a referència per mida i orientació.

Un cop preparat tot allò que es considera fix es col·locarà la càmera, evitant posicions on el robot o les columnes tapin la base ja que es perdria zona útil com també evitant punts de vista a traves de les parets de metacrilat ja que poden produir reflexos. Aquesta col·locació serà sense preocupar-se de fins on arriba l'angle de la càmera ja que un cop iniciem el codi es podrà ajustar la posició correctament amb l'ajut d'una previsualització del que està observant.

3.2. Inicialització del codi

Per tal de donar-li al codi la possibilitat de ser utilitzat en diferents llocs s'ha dissenyat un protocol d'inicialització que permet la instal·lació i parametrització de la càmera.

És important destacar que sempre s'ha d'assegurar que els 4 TAG quedin a la vista ja que sense ells no serà possible orientar-se. Per assegurar aquesta condició el codi executarà la previsualització que permetrà observar què veu la càmera en temps real. Un cop els TAG estiguin dintre de la imatge polsarem enter per tancar la previsualització i agafar una imatge.

S'obtindrà una imatge de 480x640x3, un format que es llegeix de la següent manera: files(o eix Y) x columnes (o eix X) x capes. MATLAB interpreta les imatges com matrius on cada píxel es representa amb una coordenada de fila i columna i pot estar composada d'una o varies capes depenent del format, en aquest cas es tracta d'una imatge en format uint8 on les tres capes representen el codi RGB de la imatge.

Filtrar aquesta imatge es un pas necessari per realitzar algunes operacions de manera optima però també per eliminar informació inútil (per algunes funcions) i que el codi s'executi el més ràpid possible. La primera opció és intentar treballar sempre amb la resolució mes baixa, si es redueix el numero de files o columnes es perdrà definició d'imatge fet que fa perdre precisió, però sí que es pot reduir el número de capes en el cas que el color sigui informació inútil en el procés actual. Per aquest motiu es filtrarà la imatge per a que nomes els TAG quedin a la vista. La opció mes rapida és treballar amb TAG negres i buscar-los en una imatge en escala de grisos, però en aquesta situació les ombres es poden confondre com a TAG, per això s'ha optat perquè aquests tinguin color i filtrar els píxels que formen la imatge per decidir si estan dintre o no d'un rang RGB.

Comparant cada píxel amb un rang de valors establerts es determinarà si aquests poden formar part d'un TAG o no, quedant així una imatge binaritzada on cada píxel té un valor de 0 o 1.

L'objectiu final és que nomes els píxels dels TAG tinguin com a valor 0 i que la resta tinguin un valor d'1, separant així els rellevants (de valor 0) per els següents passos o els irrellevants (de valor 1).

Un cop la imatge està binaritzada la inicialització del codi es pot donar per acabada.

3.3. **Orientació en l'espai i transformació**

Per tot el codi apareixen situacions en les que es consulten els píxels veïns. Per evitar problemes en les posicions que toquen el marge on, en cas de trobar un píxel rellevant, es consultarien una o varies coordenada inexistents, fet que provocaria un error, s'instal·larà un marc d'un píxel de gruix amb el valor 1.

Ara si, amb la imatge preparada es realitza un etiquetatge en el que cada figura que apareix a la imatge es diferencia amb un valor. Això desfà la feina feta per reduir la imatge però és un pas necessari. En aquest pas encara es pot diferenciar entre informació irrellevant indicada amb el valor 1 i la resta de valors que poden o no ser rellevants.

Un cop etiquetat es calcularà l'àrea de cada figura descartant els valors superiors a IMmax ja que es considera que cap TAG tindrà aquest valor i que equival a un conjunt de ombres o elements de fons i inferiors a IMmin que es consideren elements de brutícia i taques.

El següent pas serà detectar els TAG. Per diferenciar-los de la resta de formes s'utilitzarà la relació entre perímetre i area (dos mesures fàcils de realitzar) que donarà valors entre 0.97 i 1.03 per als cercles, ja que al ser una imatge discreta la relació no és perfecta, i entre 0.85 i 0.93 per al quadrat. Aquests valors queden lluny del que la majoria de formes retornaran però es possible que algunes

formes amorfes amb forats retornin valors semblants, per això es necessari un pas previ que elimini les formes foradades.

Un cop els TAG han sigut identificats calcularem el seu centroide aprofitant el fet de que es un punt que es veu menys afectat per la discretització i serveix com a referència d'on estan situats els TAG. Un altre conjunt de coordenades que resultarà útil és la relació entre l'amplada i la profunditat dels TAG. Com que aquesta posició i les operacions amb aquestes dades es veuen molt afectades per la discretització s'intentarà sempre utilitzar una mitjana dels resultats obtinguts amb el major numero de TAG possibles, però donant un major pes als TAG més grans ja que això implica que l'error que cometen és menor degut a que cada píxel representa una fracció menys significativa.

Utilitzant el centroide s'assignarà a cada TAG una cantonada (i a cada cantonada un TAG). És important que es compleixi la restricció anterior ja que d'aquesta manera es podran calcular els valors necessaris per tota la transformació de la imatge.

Es calcula la posició de l'horitzó de fuga utilitzant el centroide dels TAG, i ara amb les dades obtingudes dels TAG i l'horitzó de fuga es realitza una petita transformada on es representarà la posició dels TAG en cas d'estar a la mateixa altura que les peces a identificar.

Ja s'han començat a definir els paràmetres necessaris per realitzar la transformació de la imatge, però també és necessari establir els límits d'aquesta. Ha de tenir la mida justa per mantenir la informació rellevant i poder obtenir tota la informació de les figures, així que es començarà per la base. Com que totes les peces estaran situades dintre del rectangle que formen els TAG, si la base s'estableix en el vèrtex inferior d'aquest rectangle (el TAG en una fila més elevada) s'asseguraran les dues premisses anteriors.

Tal i com funciona la transformació de la perspectiva cònica (que s'explica en l'apartat4) l'amplada de les figures és la resultant de fer una projecció des de el punt de fuga (punt situat aproximadament al centre de la imatge a l'altura de l'horitzó de fuga) a la base, on es marcaran els valors extrems dels TAG. Degut a que aquesta amplada no variarà indiferentment de la inclinació de la càmera es farà servir aquesta amplada com a referència de distancia entre realitat i píxel.

Amb aquesta relació i les inclinacions respecte els eixos de coordenades obtingudes al realitzar altres operacions s'estableix la distancia que hauran de tenir els TAG entre si (nomes parlem de profunditat, la distancia de profunditat queda directament establerta per el punt de fuga i la base). Amb aquest valor ja es poden calcular la resta de punts necessaris per fer la transformació i llavors determinar la resta de límits (superior i laterals).

Es transformen les imatges (originals o amb el mínim tractament possible) de tal manera que quedarà una aproximació de com seria en vista zenital, però tot i que el rectangle que formen els TAG

encara no queda en paral·lel als eixos de coordenades de moment no el modificarem ja que no es necessari per aplicar els mètodes requerits i d'aquesta manera preservarem el màxim d'informació.

Ara els TAG passen a un segon pla fins que s'hagi de quadrar la imatge i nomes utilitzarem la informació de les figures. S'ha de trobar, etiquetar i calcular la seva area centroide com s'ha fet amb els TAG i mitjançant aquestes dades i el perímetre es calcularan relacions per establir quin element de la base de dades creada correspon amb cada figura.

4. Projecció cònica

Com s'ha exposat en els capítols anteriors s'utilitzaran les propietats de la projecció cònica per reproduir la imatge zenital de la vista de la càmera. En aquest apart s'explicarà en què consisteix.[5]

4.1. Introducció a la projecció cònica

L'objectiu d'aquesta projecció és poder representar un objecte tal i com el veiem, on depenent del punt de vista quedarà representat d'una manera o una altre. Així obtenim sensació de profunditat, o dit d'una altre manera, com la figura de la dreta queda representada en una posició real com la de l'esquerra.

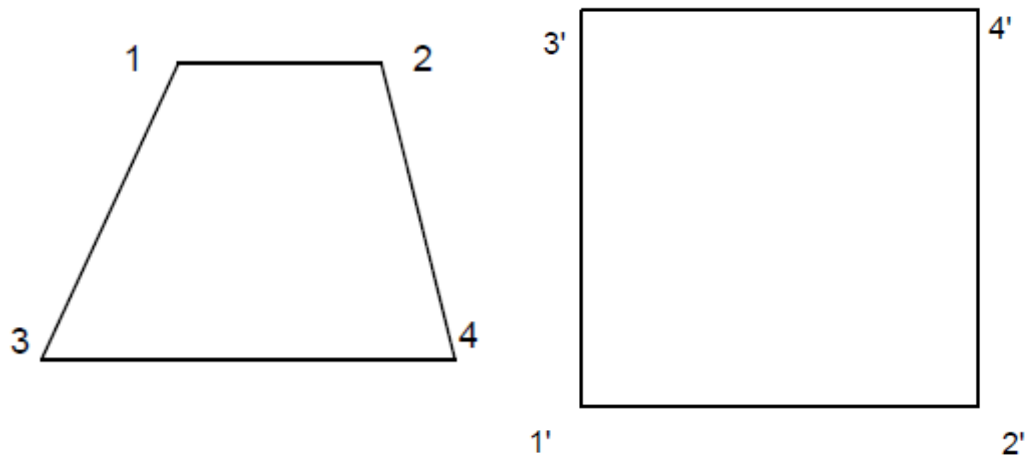

Figura4.1. Un rectangle representat en una projecció cònica i vista zenital

Primer s'explicarà què és i en què consisteix aquest sistema de representació. A diferencia de les projeccions paral·leles, la central o cònica no mante la distancia entre els punts en dos línees paral·leles, aquestes tal i com ho veiem els humans acaben coincidint en l'horitzó i sempre serà així a excepció de que aquestes línees siguin paral·leles respecte l'horitzó.

Per entendre la projecció cònica són necessaris 3 elements:

- El punt o punts de fuga:són els punts del pla auxiliar on les rectes paral·leles concorren.
- El pla auxiliar: és el pla on es projecten les imatges, en aquest cas la pantalla.

- El punt de vista: és la posició de l'observador, en aquest cas la càmera. En aquest document s'anomenarà Altura, cal aclarir que degut a que el punt de vista és una càmera aquest sempre estarà centrat en l'eix X i per tant només ens preocuparem de la seva posició en l'eix Y.

També cal d'estacar altres elements importants com:

- La línea d'horitzó: línea imaginaria on es situen tots els punts de fuga. En aquest document s'anomenarà Y_Fuga.

Per realitzar aquesta projecció s'utilitzen diferents mètodes depenent del punt de vista i de l'objecte a transformar, si és un objecte vertical o horitzontal, com que la selecció es realitzarà segons la forma de la superfície superior ens centrarem en la transformació de figures en posició horitzontal o sobre el pla.

4.2. Projecció cònica sobre el pla a zenital

4.2.1. Figura en orientació senzilla

En aquest apartat s'explicarà el procés que es segueix per trobar els punts des de una projecció cònica sobre el pla a una zenital. Per fer això es començarà amb una imatge senzilla on el punt de fuga de les línees paral·leles coincideixi amb el punt de fuga de la transformació.

La projecció cònica sobre el pla és una perspectiva frontal o amb un sol punt de fuga, centrada en aprofitar les característiques que tenen els elements que comparteixen altura.

Pas 1: Es tracen dues línees que segueixen els laterals de la nostra figura fins que concorren com es pot veure a la Figura 4.2, en aquest punt es pot establir la línea de l'horitzó de fuga i, en aquest cas, també el punt de fuga.

El punt de fuga de la figura coincideix amb el de la imatge perquè la primera es troba en paral·lel amb la línea d'horitzó, fet que es pot observar ja que les línees superior i inferior són completament paral·leles i tot i que la figura no es troba centrada en l'eix X (cosa que es pot observar ja que les línees de projecció no formen angles conjugats) això no afecta a aquesta coincidència.

També establim la base de la imatge, la seva col·locació no te un criteri geomètric, per això se n'utilitzarà un d'eficiència i situarem la línea de base a partir del punt en el que ja no tinguem més informació útil.

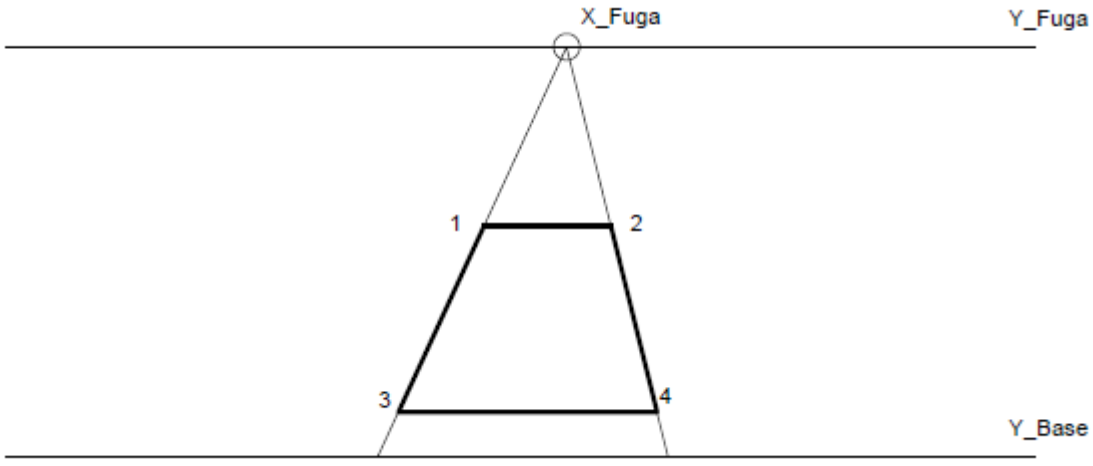

Figura 4.2. Representació del primer pas per transformar la imatge.

Pas 2: Com s'observa en la Figura 4.3els punts referents a l'altura (X_Altura i X_Alturae) són simètrics respecte el punt X_Fuga. Nomes s'explicarà el mètode per situar els punts 1 i 3.

Al projectar els punts1 (que també s'anomenarà a) i 3 (que també s'anomenarà b) a l'horitzó de base apareix el punt xβ i en realitzar la mateixa operació amb els punts de l'altre banda s'obté la distancia d, aquesta és la distancia que tindran els punts 1 i 2 entre ells, de la mateixa manera els punts 3 i 4.

Com que la figura final és un quadrat, la distancia de projecció entre δa i δbha de equivaldre a d. Aquesta restricció es podrà entendre millor en el pas 3, a més de concorre a l'altura de Y_Fuga. Un cop es te X_Altura o X_Alturae simplement es realitza la simetria en el punt X_Fuga.

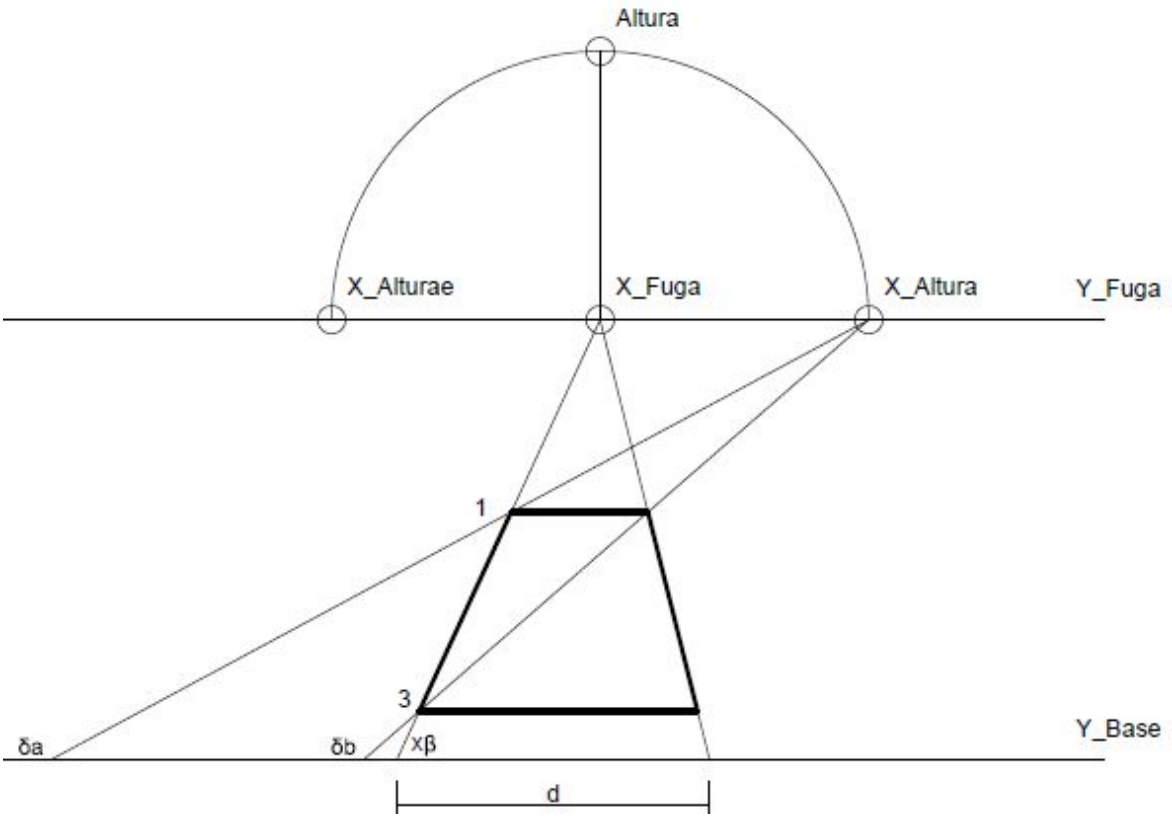

Figura 4.3. Representació del segon pas per transformar la imatge.

Pas 3: És el pas final, un cop es tenen els punts d'Altura, Fuga i Base la transformació queda parametritzada, i només queda buscar cada punt. Com es pot veure a la Figura 4.4el procés és senzill, després d'obtenir els punt β i δ es traçarà un arc de π/2 radiants des de el punt β amb un radi equivalent a la diferencia entre els dos punts i es repetirà aquest procés per cada punt de la imatge original (Figura 4.5).

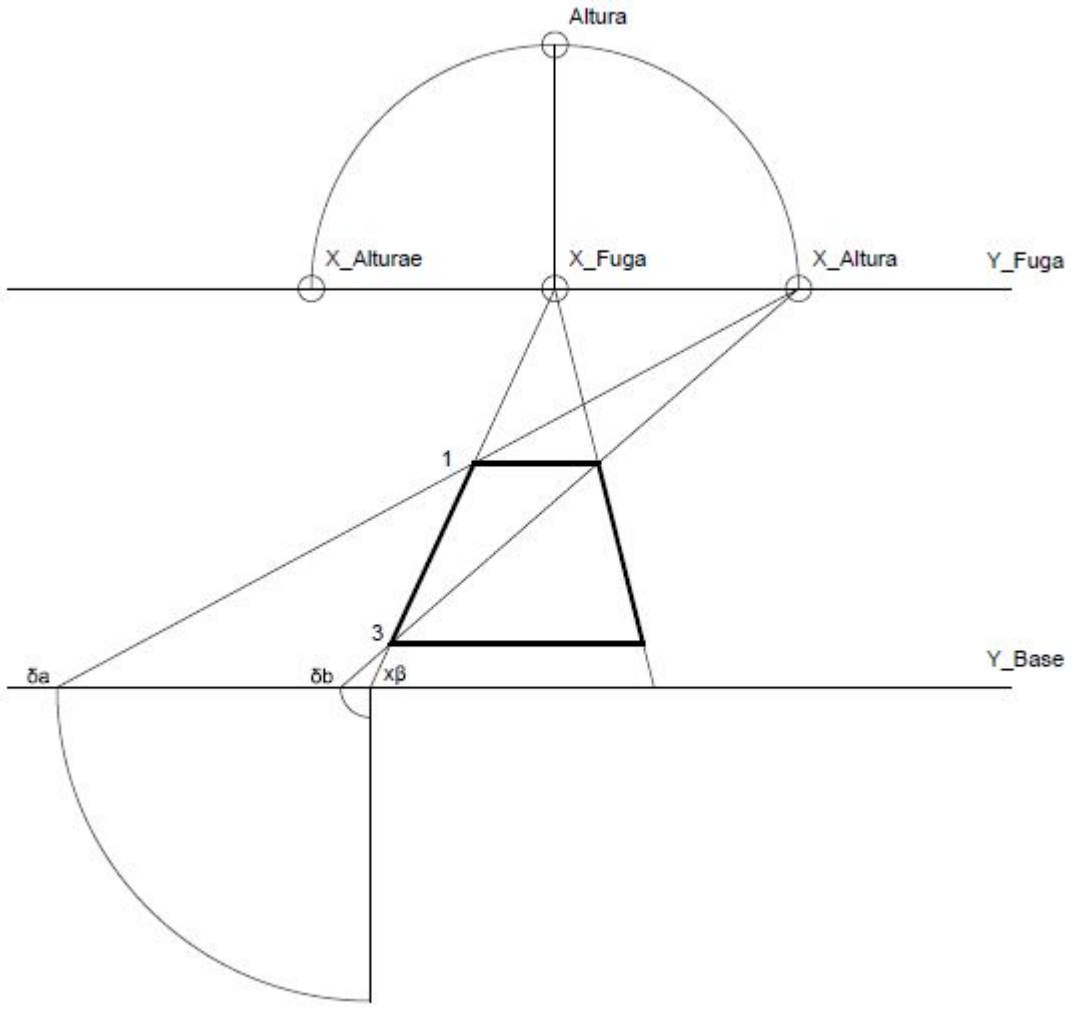

Figura 4.4. Representació del tercer pas per transformar la imatge.

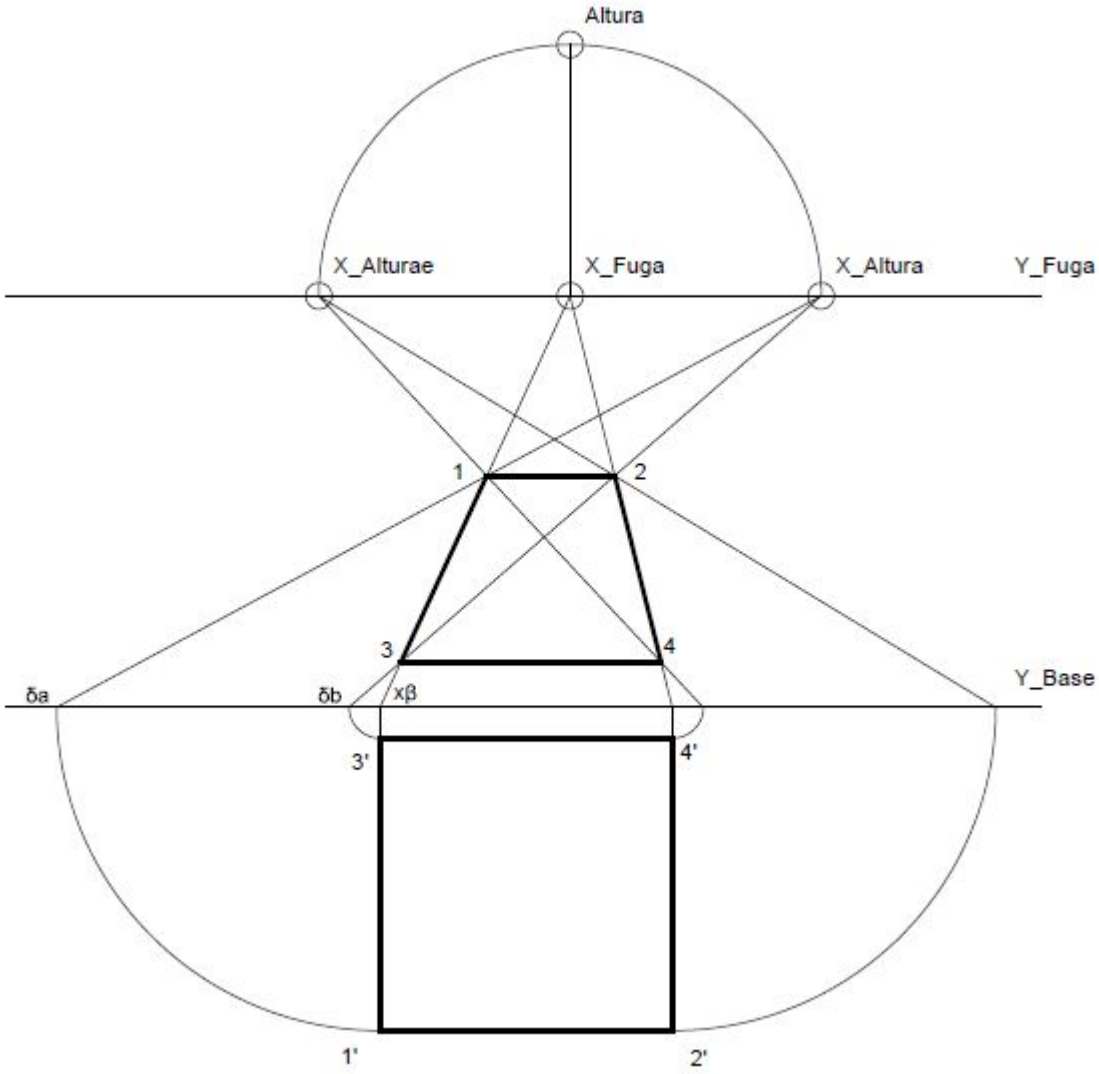

Figura 4.5. Representació completa de la transformació.

Un cop explicat amb aquesta figura senzilla s'exposarà un cas més pròxim al que ens podrem trobar en executar el codi.

4.2.2. Figura en orientació complexa

Com es pot observar en la Figura 4.6algunes de les propietats que tenia la figura anterior ja no es compleixen i per això el procés es complica una mica.

Pas 1: Com en el cas anterior el primer que es fa és projectar les línees laterals cap a l'infinit i allà on concorren s'estableix l'horitzó de fuga (Y_Fuga), en aquest cas la figura no està posicionada paral·lelament al'horitzó, és per això que el punt on concorren les dues línees no correspon amb el punt de fuga utilitzat (X_Fuga). De la mateixa manera que en el cas anterior l'horitzó de base s'estableix en el punt en el que la informació deixa de ser útil.

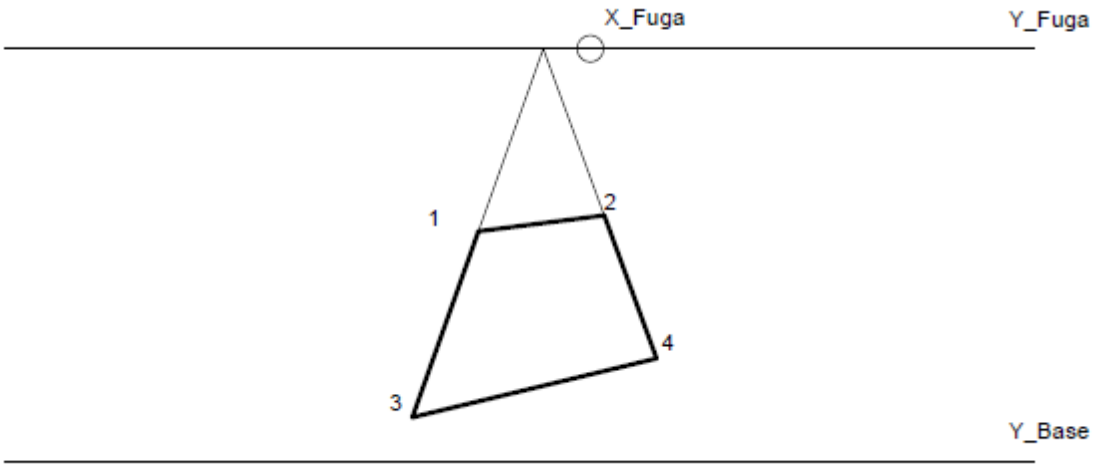

Figura 4.6. Representació del primer pas amb una figura en orientació complexa.

Pas 2: Al repetir el pas 2 del apartat 4.2.1 es pot observar en la Figura 4.7 com en aquest cas no apareix una xβ al projectar les línees desde X_Fuga a Y_Base passant per els punts 1 i 3,sinó que ens apareixen dos punts que anomenarem βa i βb. També es pot observar com apareixen dues parelles de punts entre les que es pot extreure d. Els dos punts han de ser de la part superior o inferior però no un de cada.

La orientació també afecta la distancia que ha de tenir δa i δb. Aquesta no serà directament d, sinó que la diferencià entre les distancies $\overrightarrow{\delta b \beta b}$ i $\overrightarrow{\delta a \beta a}$serà igual a d. Aquest fet es pot observar amb la Figura 4.8on, al realitzar el pas 3, la distancia entre els arcs coincideix amb d i les distancies $\overrightarrow{\delta b \beta b}$ i $\overrightarrow{\delta a \beta a}$ equivalen als punts finals de cada arc.

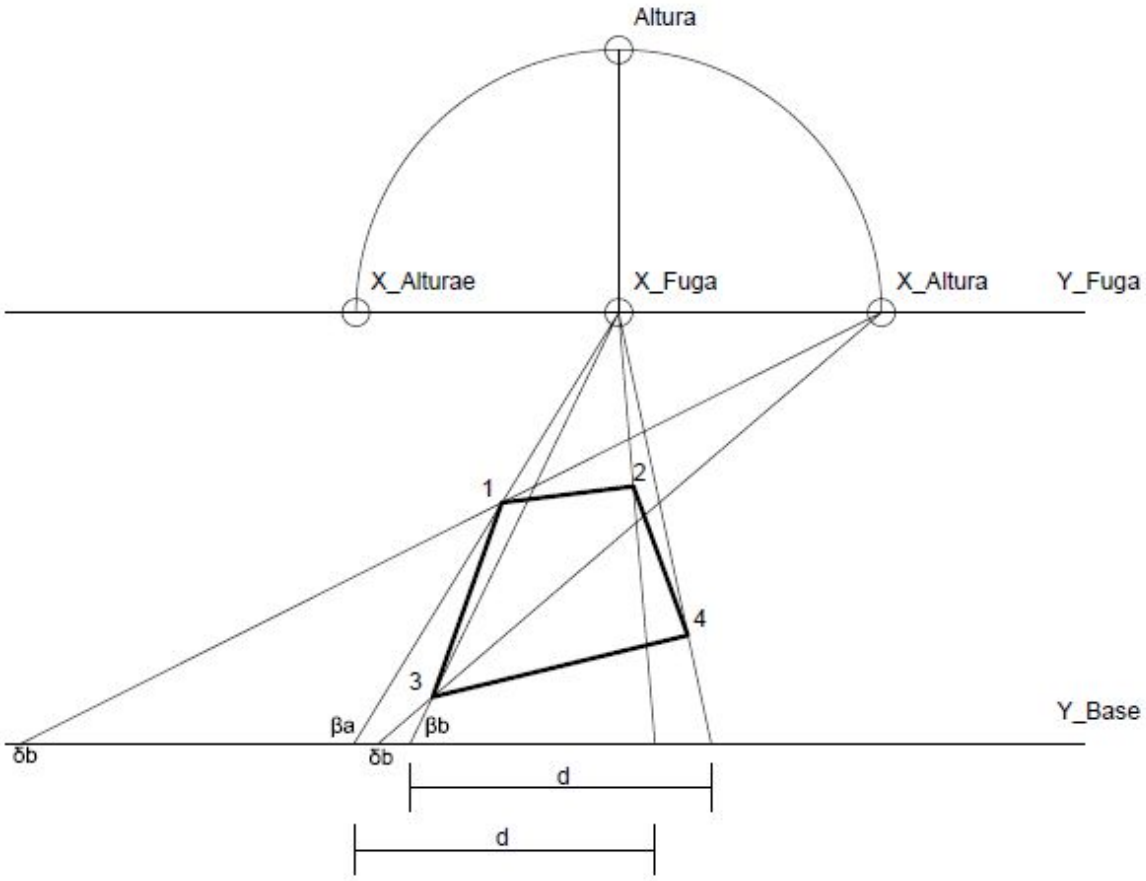

Figura 4.7. Representació del segon pas amb una figura en orientació complexa.

Pas 3: Aquest pas es igual al realitzat en el apartat 5.2.1

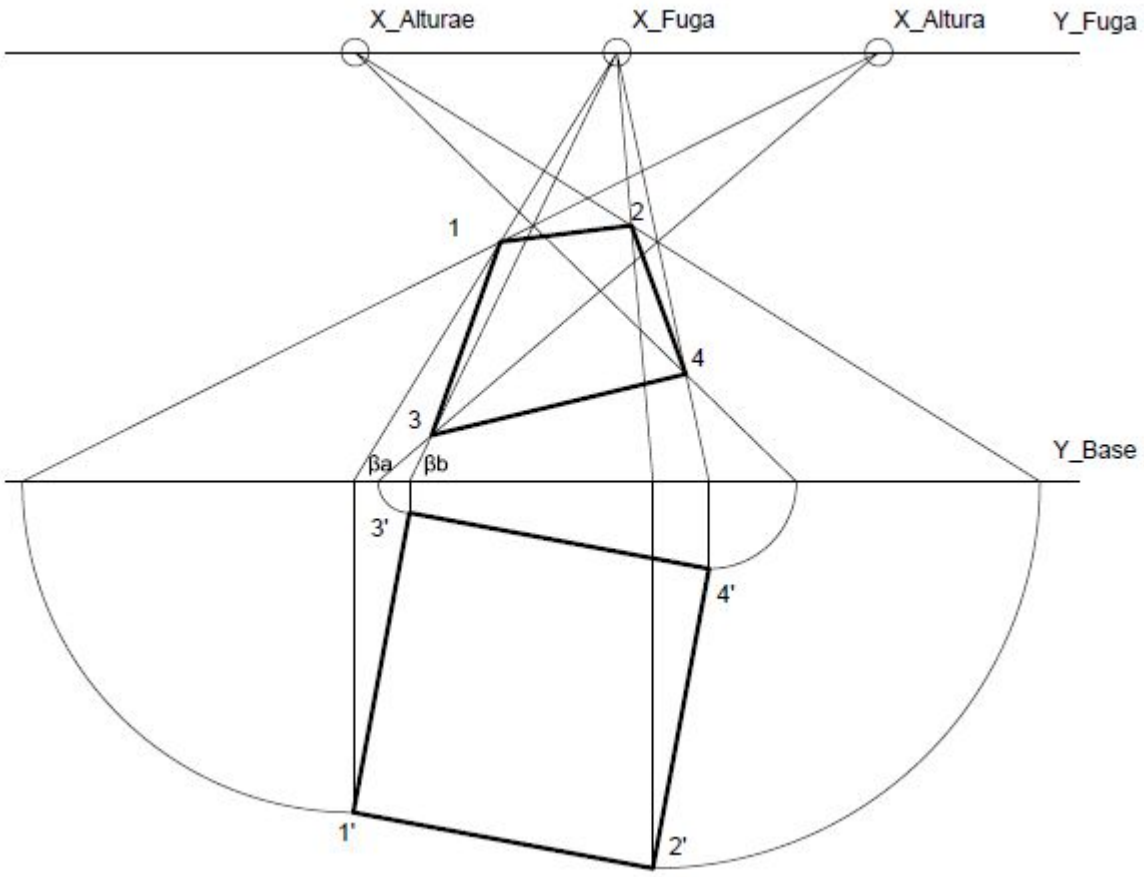

Figura 4.8. Representació del tercer pas i figura completa amb una figura en orientació complexa.

5. Codi

En aquest capítol s'explicarà el codi dissenyat de forma detallada amb explicació matemàtica de les equacions importants que hi apareixen.

El codi s'ha dissenyat amb una estructura que permeti entendre què realitza el codi nomes observant la funció Main i entendre com realitza cada acció observant la funció concreta. El repartiment del codi en funcions dona una imatge mes clara i neta d'aquest.

L'ordre que es seguirà per explicar les funcions segueix l'ordre de aparició dintre del codi.

5.1. **Main**

La funció Main o principal és la funció que s'executarà en inicialitzar el codi. Aquesta s'encarregarà de comunicar-se amb l'usuari per inicialitzar el codi o seleccionar quina informació volem i de cridar a la resta de funcions en el moment necessari a excepció de la funció AlgTortugaPapert que es cridarà dintre de la funció AlgEtiquetatge.

Un cop s'introdueixen automàticament alguns paràmetres generals i la informació de les figures el primer que realitza el codi és la captació de la imatge amb la que es treballarà. Com s'ha mencionat en el apartat 3.2 és necessari que en la imatge captada es puguin veure els 4 TAG, per això apareixerà una finestra on podrem veure en temps real què està captant la càmera. Un cop centrada caldrà polsar enter per acabar la previsualització.

Al acabar la previsualització obté la imatge Figura 5.2 amb l'objectiu de recuperar la informació de les peces com seria a Figura 5.1. Gracies al calibratge (capítol 6) obtenim la imatge Figura 5.3 on la distorsió de la lent presumiblement ja no afectarà la imatge.

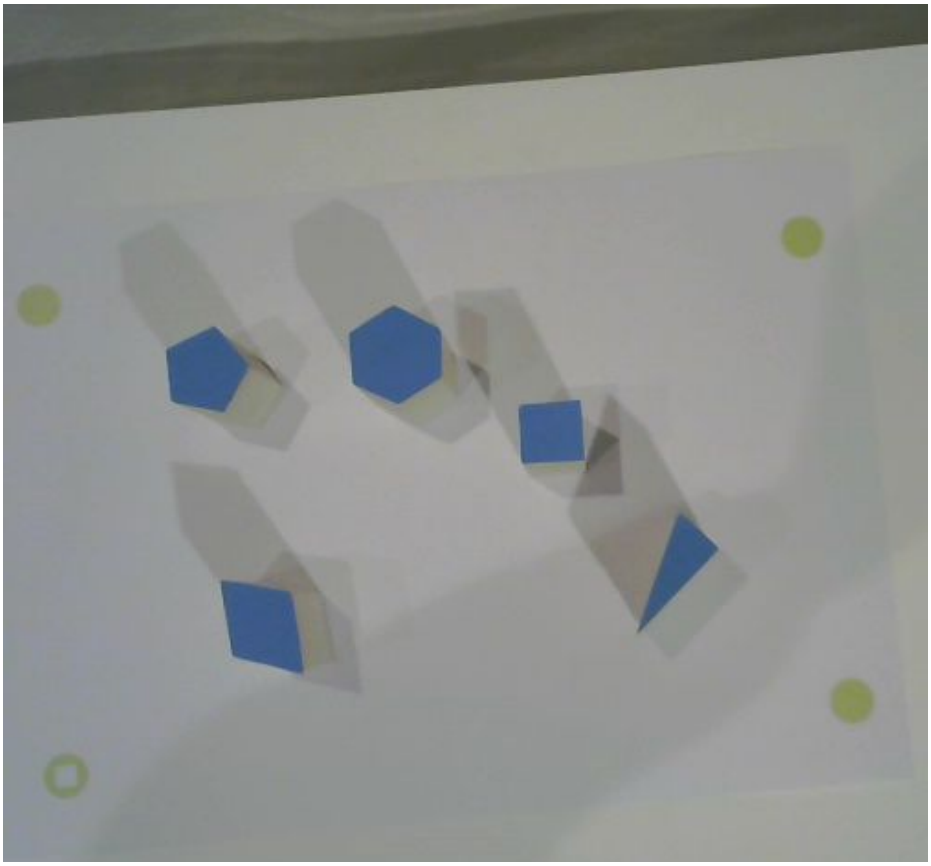

Figura 5.1. **Vista zenital real de les imatges que s'utilitzaran per demostrar els passos.**

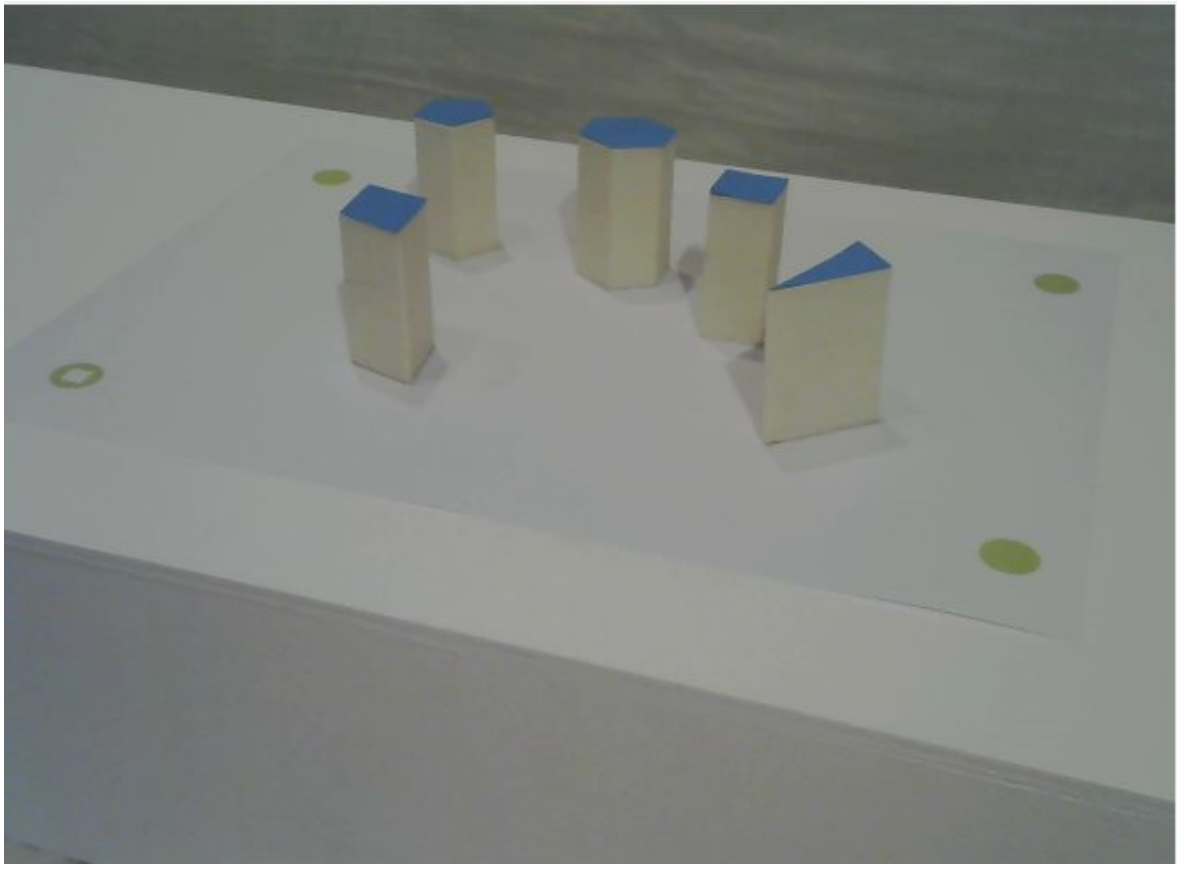

Figura 5.2. **Imatge captada, base de les imatges de mostra, im.**

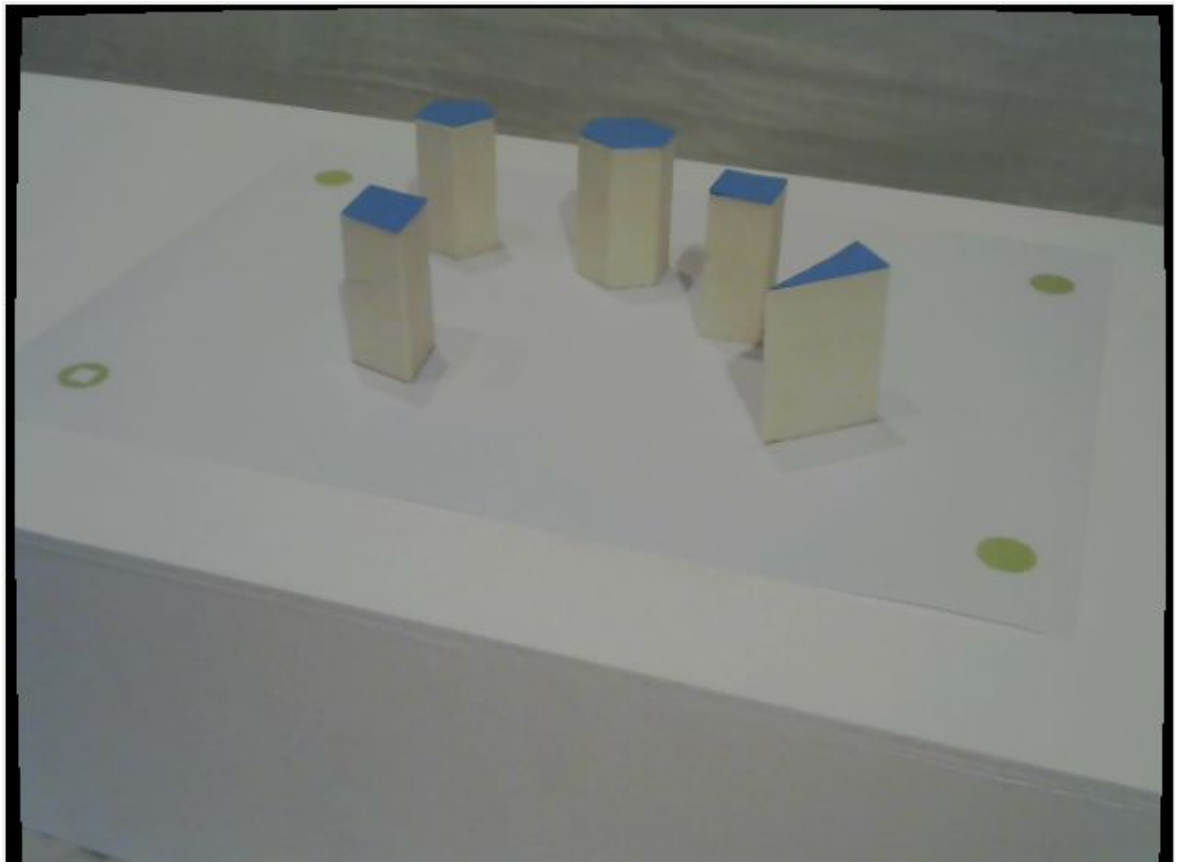

Figura 5.3. **Imatge modificada segons el calibratge de la camera.**

Un cop acabat l'anterior apartat ja tindrem la imatge im en blanc i negre Figura 5.4 (IM10) on les siluetes que ens interessen quedaran en negre. Per poder identificar els TAG el codi etiquetarà amb un valor diferent cada silueta que hi apareixi mitjançant la funció Etiquetatge (apartat 5.3). El resultat final quedarà relaxat a IM11 i el numero de siluetes diferents (contant el fonts blanc) quedarà relaxat a k.

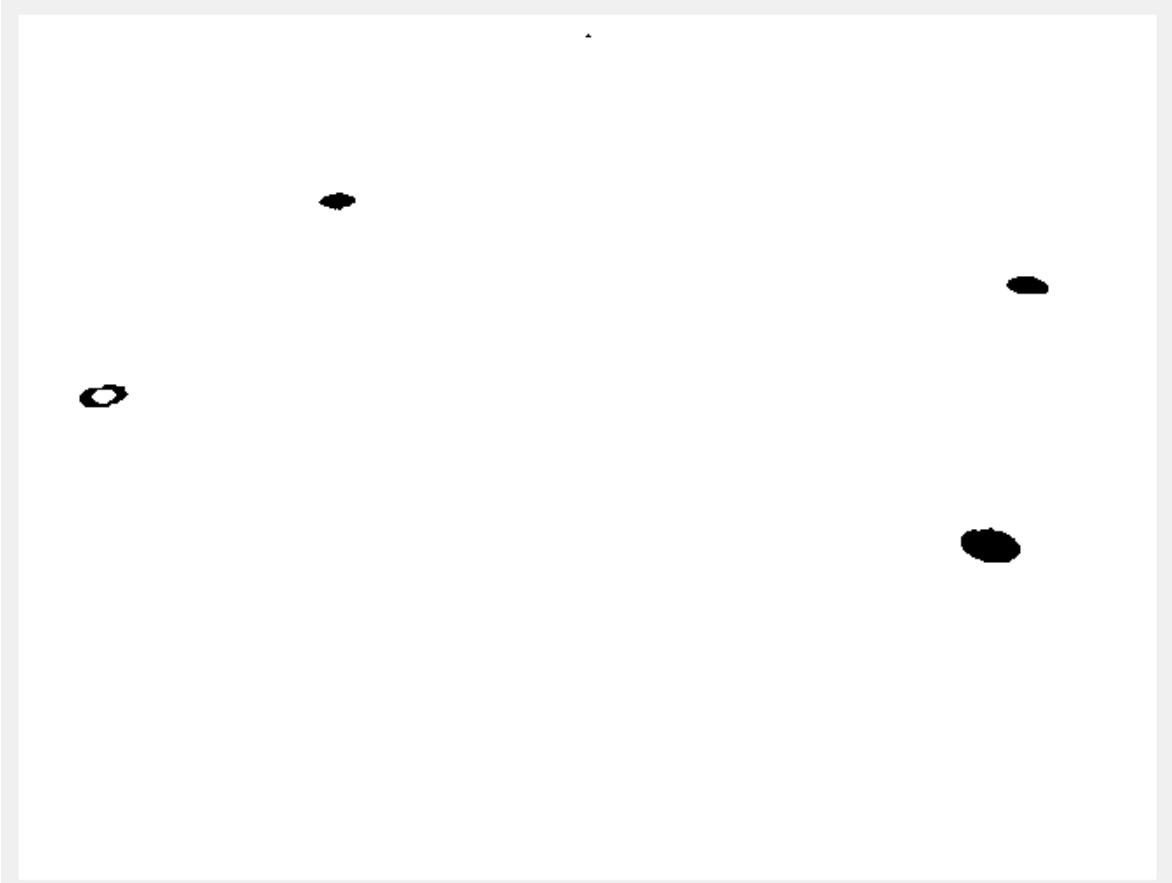

Figura 5.4. **Imatge segmentada on només es destaquen els possibles TAG.**

Amb les siluetes diferenciades es poden començar a realitzar filtres més concrets. El primer serà repàs a les siluetes unificant les que estiguin en contacte com s'explica a l'apartat **¡Error! No se encuentra el origen de la referencia.** on també es determina quin TAG és el quadrat i després un filtre segons l'àrea (karea). Primer esborrem les siluetes mes grans o mes petites que els paràmetres preestablerts que limiten l'àrea màxima i mínima que pot tenir un TAG dintre del rang de treball, això eliminarà siluetes de fons i també petits grups de píxels que podem considerar brutícia otorgant el valor d'1 a cada píxel afectat. El resultat d'aquesta operació s'introduirà a IM12.

En aquest punt tant el vector que recull tant les identificacions útils com les àrees conté molts espais buits, per això es realitzaran de nou els vectors obviant els 0 però guardant l'ordre per tal de mantenir una correspondència en la posició entre les àrees del vector Area_IM12 i les identificacions Elements_IM12.

Executant les funcions AlgCercles, CalculCentre, Eix, AlgCantonades (apartats 5.6, 5.7, 5.8 i 5.9 respectivament) s'identificaran les siluetes que corresponen a TAG, recordant quin és el quadrat, a quina cantonada corresponen i la informació necessària per identificar a quina distancia i orientació es troba el pla de treball.

Aplicant la tècnica explicada en l'apartat 4.2.2 es calcula la posició del punt de fuga amb la funció AlgFuga (apartat 5.10). Aquests punts són sempre els mateixos independentment de l'altura del pla, però no els punts del al pas 3 corresponent a la altura, ja que aquests varien segons l'altura del pla i no es té cap interès en transformar la imatge a l'altura de la superfície dels TAG.

Es calcula una aproximació de quina es la inclinació de l'altura del pla on es troben les cares superiors dels objectes (que anomenarem figures) coneixent la relació entre aquest i l'amplada dels TAG, i es projecta el que seria la superfície dels TAG en cas d'estar a la mateixa altura que les figures amb la funció Abatirplano (apartat 5.11), quedant guardat en la imatge IM50. Això farà que la posició dels TAG guardada a PosY quedi obsoleta, per aquest motiu s'obtindrà novament la seva posició en l'eix Y i es desarà a PosY_Elevat amb la funció AlgAltura.

Ara si, amb els TAG a una altura interessant es comencen a obtenir la resta d'informacions necessàries per realitzar el pas 3. Amb la funció AlgBase (aparat) determinem el límit inferior Y_base i amb la funció AlgRelacioPixelRealitat (apartat) obtenim informació necessària per determinar quina ha de ser la distancia. Aquesta variarà segons la inclinació que tingui la estructura dels TAG respecte l'horitzó de la imatge, per això cridem la funció OrientacioA (apartat 5.14). Aquesta funció ens retorna la orientació des de la que veiem la estructura dels TAG i la D obtenint ja tots els paràmetres necessaris per el càlcul dels punts finals necessaris per fer la transformació X_altura i X_alturae.

Per assegurar-se de que tota la informació útil quedarà dintre de la imatge final s'identifica la zona amb la funció AlgLimitZonaTreball (apartat 5.16) i es transformen les imatges. Aplicant la funció AlgTrans (apartat 5.17) dos vegades, obtindrem la imatge transformada de IM50 i IM140 (Figura 5.5), la primera necessària per quadrar la estructura dels TAG amb els eixos de la imatge i la segona per realitzar totes les operacions d'identificació.

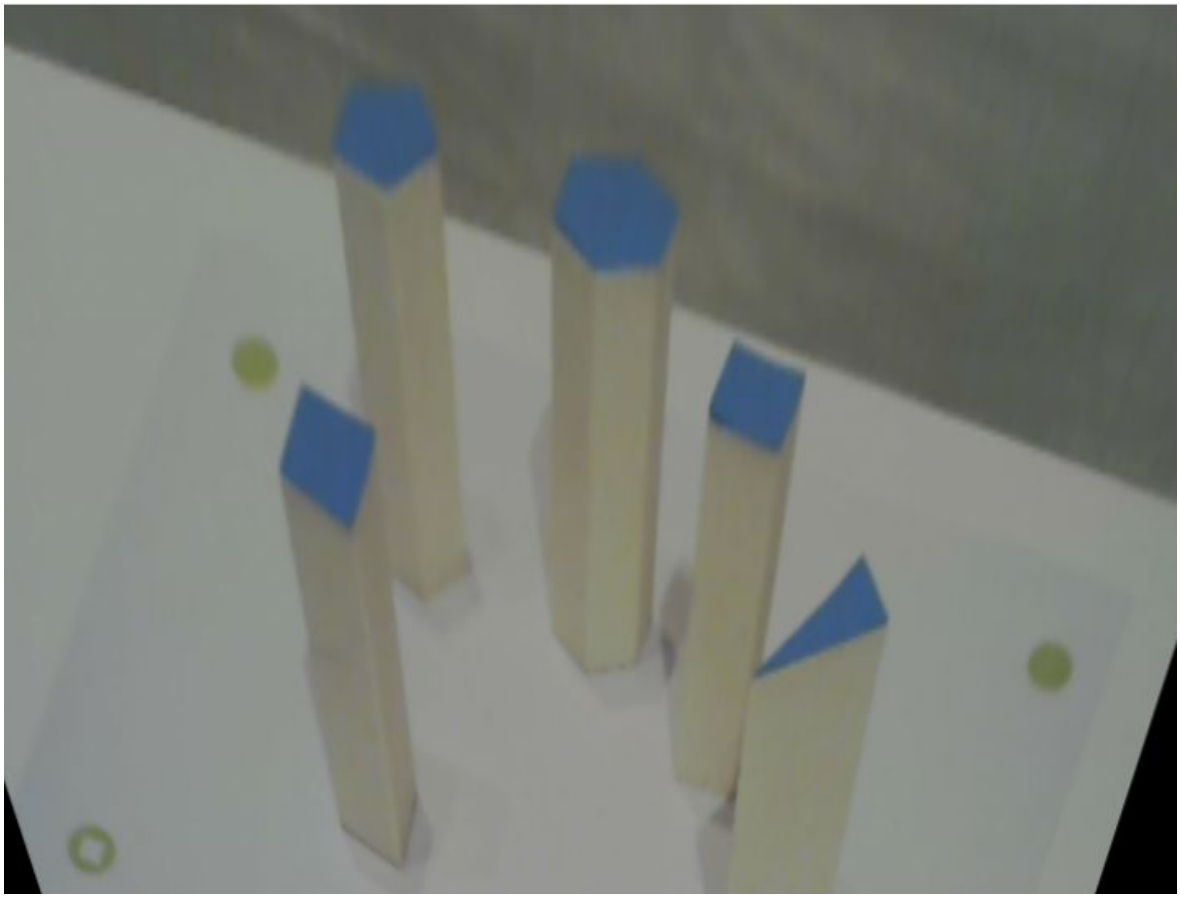

Figura 5.5. **Imatge transformada on ja es poden aprecia les siluetes reals de les peces.**

Per a la identificació de figures es repetirà la seqüència realitzada per identificar els TAG, però la preparació prèvia consistirà en un filtre de colors i s'eliminaran les petites protuberàncies amb un disc de radi 2 i la funció de MATLAB imclose.

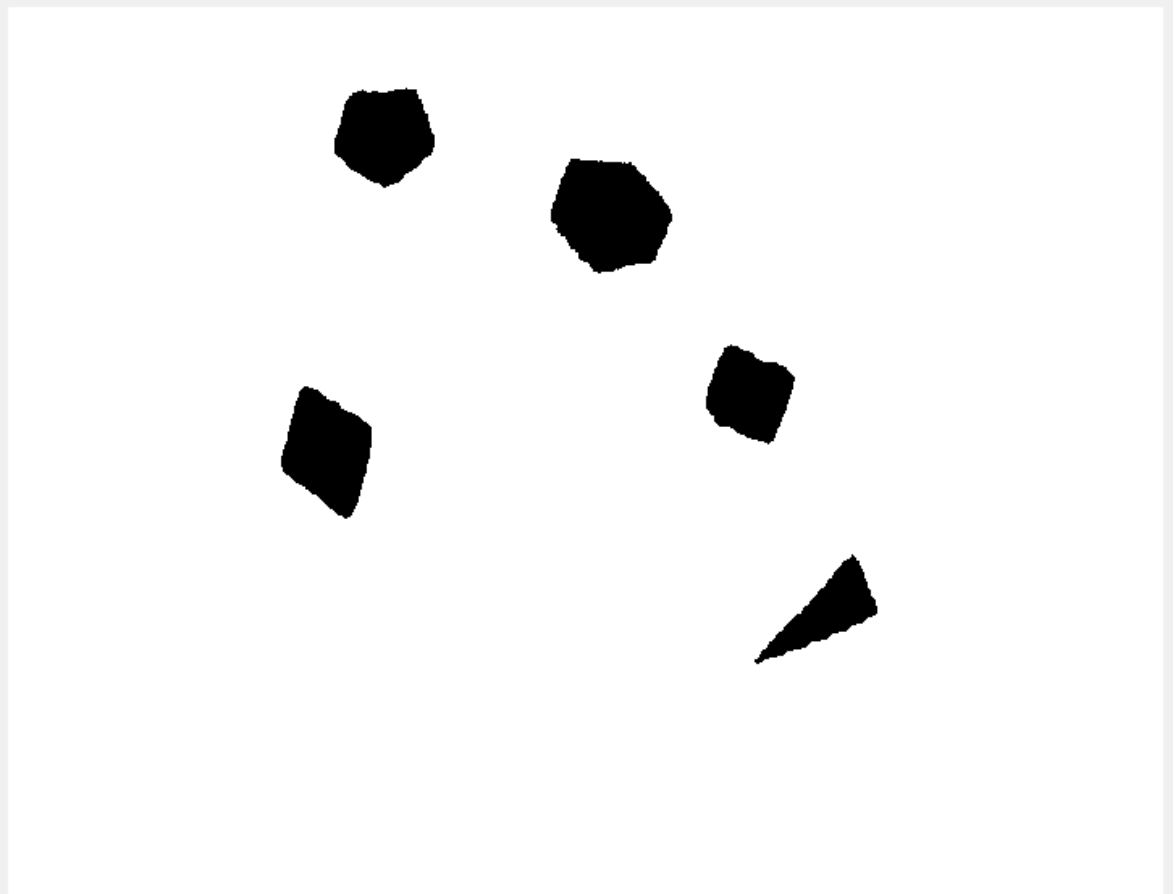

Figura 5.6. **Imatge segmentada on s'observen les figures amb una silueta real.**

Després d'etiquetar, calcular àrees i el centre de les figures es començarà amb la obtenció d'informació per identificar cada peça. Amb la funció Marges (Apartat 5.20) es separarà el marge de les figures de la resta del cos assignant el valor 0 a l'interior. Amb la funcio AlgVertex (Apartat 5.21) aprofitant la imatge IM157 que ha tornat la funció Marges es buscaran els vèrtexs de cada figura i amb la informació d'aquests la funció Identificació (apartat 5.22)decidirà quin nom correspon a cada figura.

L'últim pas abans de poder interactuar amb la informació obtinguda és colocar finalment les figures en la posició en la realitat respecte el punt origen (que correspon al centre del TAG del quadrat). D'això se n'encarregaran les funcions AlgOrientacioB (apartat 5.23)i RotacioPosicions (apartat 5.24), la primera retornarà quant s'ha de girar per a orientar els eixos a la realitat i la segona executarà el reposicionament quedant relaxat en els vectors PosX_Final i PosY_Final.

Finalment el codi es comunicarà amb l'usuari per la consola indicant-li quines figures ha detectat Figura 5.7.

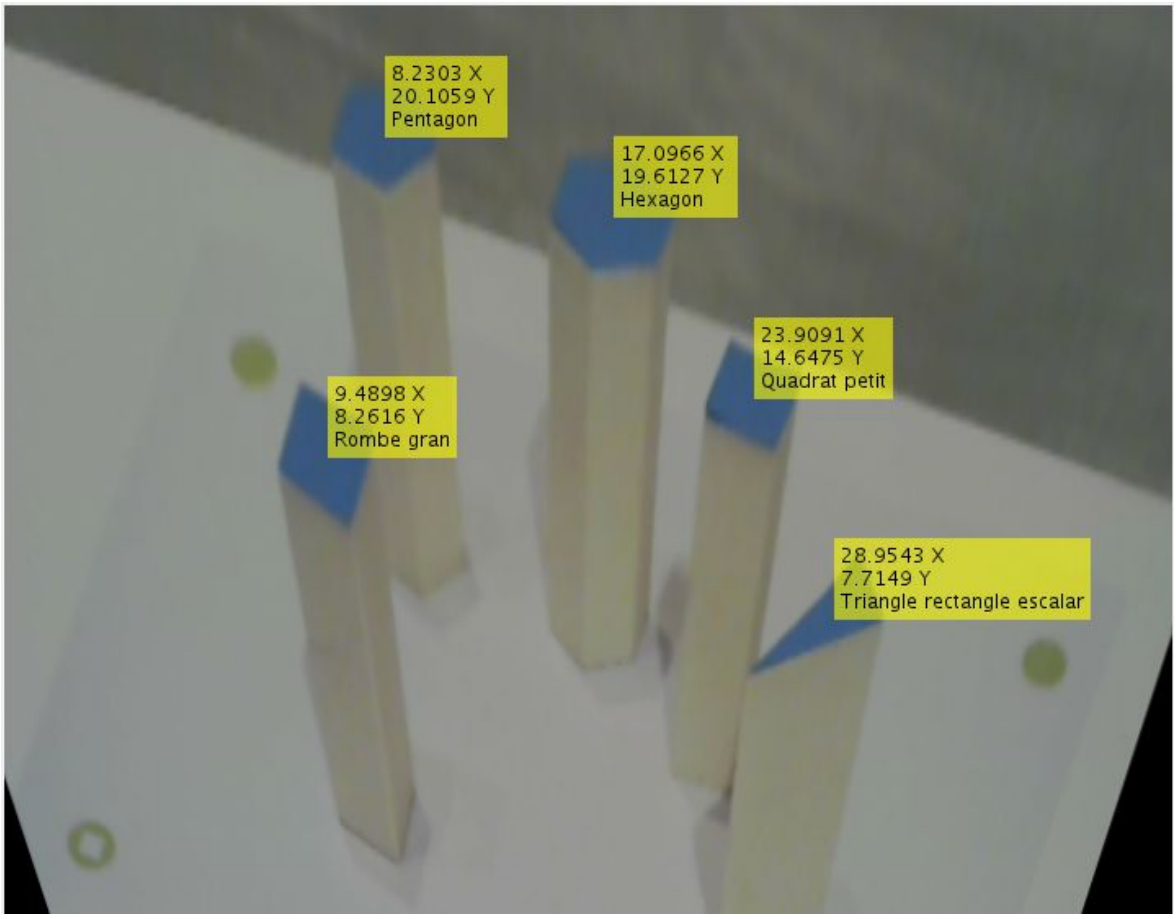

Figura 5.7. **Imatge transformada amb les etiquetes de les peces.**

5.1.1. Pseudocodi

Inicialització de constants
Activació càmera
Previsualització en vídeo
im=Captura d'imatge
im=Correcció calibratge càmera
Per dos vegades
    Si és segona volta
        Rotar imatge segons inclinació
    Fi Si
    IM10=Segmentació im per trobar TAG
    IM11=Etiquetatge (apartat 5.3)
    Càlcul area etiquetes
    Eliminar àrees amb etiquetes grans o petites
    Detecció de TAG (apartat 5.6)
    Identificació TAG de referència(apartat 5.6)
    Càlcul centre de gravetat TAG (apartat 5.7)
    Identificació inclinació angle alfa d'Euler (apartat 5.8)
    Assignació correspondència TAG amb cantonades (apartat 5.9)
    Càlcul inclinació pla figures (apartat 5.10)
    Projecció etiquetes TAG al pla de les figures (apartat 5.11)
    Assignació límit inferior zona útil (apartat 5.12)
    Càlcul centre de gravetat TAG projectats
    Identificació relació píxel amb realitat (apartat 5.13)
    Càlcul distancia D (apartat 5.14)
    Càlcul altura càmera (apartat 5.15)
    Càlcul inclinació angle gamma d'Euler
Fi Per
Assignació límits superior, esquerra, dret (apartat 5.16)
IM50=Transformació imatge a perspectiva zenital (apartat 5.17)
Segmentació IM50 per trobar figures
Etiquetatge figures
Càlcul area figures (apartat 5.19)
Càlcul centre de gravetat figures
IM150= Marges figures IM50 (apartat 5.20)
Identificació vèrtex figures (apartat 5.21)
Identificar tipus de figura (apartat 5.22)
Obtenció posicions figures respecte TAG referència (apartat 5.23)
Rotar posicions figures per quadrar amb eixos X Y (apartat 5.24)
Mostrar posicionament real

5.2. **TrobarTagColor**

La funció TrobarTagColorsegmenta la imatge separant la informació útil de la inútil, en aquest cas separa els píxels que per codi de colors RGB poden ser considerats TAG de la resta.

Per fer aquesta funció només necessita la imatge i les dimensions d'aquesta, i torna la imatge ja segmentada. El procés és simple; repassa cada píxel i comprova si compleix amb les relacions entre els valors RGB.

5.2.1. Pseudocodi

```
funció (IM_Out)= FuncioTrobarTag(IM_In,DimIM)
sortides
IM_Out          Imatge segmentada
entrades
IM_In           Imatge d'entrada
DimIM           Vector amb les mides de IM_In
Per tot X de IM_In
        Per tot Y de IM_In
                Si  valor IM_In(posició) és verd
                        IM_Out(posició)=1
                Sinó
                        IM_Out(posició)=0
                Fi Si
        Fi Per
Fi Per
```

5.3. **AlgEtiquetatge**

La funció etiquetatge separa les siluetes assignant valors diferents a cada unasegons apareixen en la imatge d'entrada IM_In i plasmant aquest resultat a IM_Out. Per realitzar-hoes fa úsd'un procés que pot generar problemes en cas detrobar-se amb una silueta en els marges, per això el primer pas és replicar IM_In amb un marc de un píxel de gruix,així amb la imatge ja preparada busca les siluetes realitzant una sèrie de consultes píxel a píxel.

La estructura de for que s'encarrega d'això consulta si la posició actual té una silueta sense identificar o no, en el cas de tenir-la consultarà si (de ser possible) en les posicions anteriors (en les dos direccions) hi ha un píxel etiquetat, ja que de ser així han de compartir el mateix identificador. En cas de no tenir cap píxel veí etiquetat voldrà dir que s'ha trobat amb una silueta nova i per tant s'iniciarà la funció AlgTortPapert (apartat5.3.1) que recorre el contorn de la silueta per deixar-lo tot amb la mateixa identificació i així, en tornar a trobar un píxel no identificat de la mateixa silueta,al consultar els seus veïns se'l identificarà amb el mateix número. Aquest algoritme permet etiquetar tota la

imatge correctament d'una sola passada. Cada cop que es troba una nova silueta s'incrementa el valor de k (dintre de la funció AlgTortPapert) ja que aquesta variable indicarà quina serà la identificació de la següent silueta (variable inicialitzada en 2 perquè el valor 1 quedarà assignat al fons de la imatge).

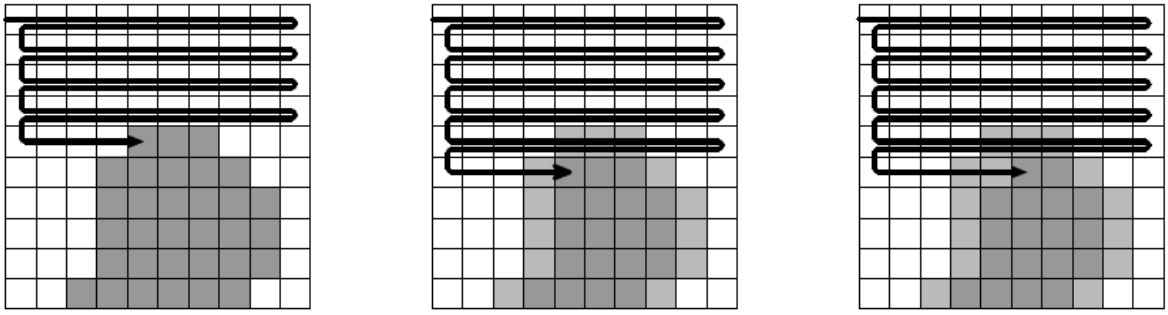

Figura 5.8.**Representació de l'etiquetatge, en la figura a) es mostra el moment en el que s'executarà la funció AlgTortPapert, en la figura b) representa la consulta d'un píxel amb un veí ja assignat i la figura c) es el pas següent on es repetiria el mateix.**

5.3.1. Pseudocodi

Funció (IM,valorSilueta)=AlgEtiquetatge(IM,DimIM)
Sortides
ValorSilueta Variable que indica el valor de la següent nova silueta
Entrades i Sortides:
IM: Imatge d'entrada i sortida, es modificarà en l'algoritme
Entrades:
DimIM Vector amb dimensions de IM
Per tota X de IM
    Per tota Y de IM
        Si valor posició actual diferent a 1
            Si valor posició esquerra te valor diferent a 1
                Valor posició actual = valor posició esquerra
            Sinó Si posició superior te valor diferent a 1
                Valor posició actual = valor posició superior
            Sinó
                Executar FunciónAlgTortugaPapert
            Fi Si
        Fi Si
    Fi Per
Fi Per

5.4. **AlgTortPapert**

La funció AlgTortPapert o Algoritme tortuga de Papert és un algoritme basat en l'algoritme de Papert, el qual localitza el contorn d'una figura.

En aquest cas la funció necessita rebre la informació de en quina fila i columna es trobava la funció que la ha cridat. Ho rebrà per f i c respectivament així com el valor que hauràd'assignar als píxels i la imatge que haurà de recorre i modificar k i IM10. Aquest últims també els tornarà com a resultat.

L'algoritme basa els seus moviments en la primera fila de la matriu MATtor que indica quin és el següent moviment. En aquesta fila hi ha dos elements, el primer indica el moviment de l'eix Y i el segon el de l'eix X. El moviment del següent pas es modificarà rotant la matriu en comptes de modificar un punter.

L'algoritme es repetirà fins que retornem a la posició inicial, la posició anterior a trobar la figura. Per saber això abans de modificar les posicions es guardaran el punt de partida a fi i ci (fila inicial i columna inicial) i es modificarà fa i ca (fila actual i columna actual)

Al finalitzar l'algoritme s'incrementarà el valor de k.

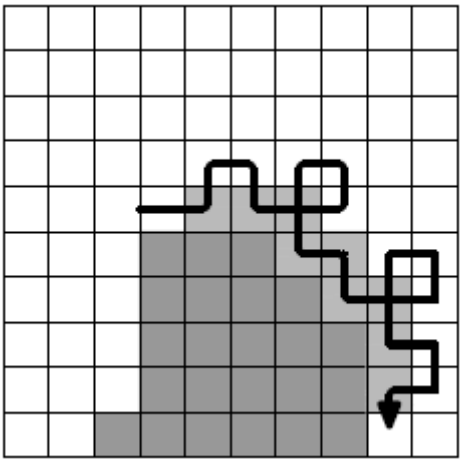

Figura 5.9.**Representació de la seqüència que seguirà l'algoritme**

5.4.1. Pseudocodi

Funció (IM,ValorSilueta)=AlgTortPapert(IM,ValorSilueta,X_i,Y_i)
Entrades i Sortides:
IM: Imatge d'entrada i sortida, es modificarà en l'algoritme
ValorSilueta Variable que indica el valor de la següent nova silueta
Nomes entrades:
X_i, Valor punter en posició d'entrada en eix X
Y_i Valor punter en posició d'entrada en eix Y
Mentre X Y actuals siguin diferents a X_i,$Y_i$repetir
Si valor posició actual es diferent a valor fons
Assignar ValorSilueta a IM(X,Y)
Desplaçar punter a la esquerra respecte l'últim moviment
Sinó
Desplaçar punter a la dreta respecte el seu últim moviment
Fi Si
Fi mentre
IncrementarValorSilueta

5.5. AlgEtiquetatge2

Continuant amb la operació que es realitzava en l'anterior apartat, aquesta funció acaba d'etiquetar els objectes que arriben en la imatge d'entrada.

Degut a com està dissenyat el codi, després de ser utilitzada la funció AlgTortPapert no tots els contorns tenen una identificació homogènia. Els contorns amb forats tindran dues identificacions o més i, aprofitant aquesta característica, es fa servir per identificar el TAG anomenat quadrat.

La funció realitza un repàs per tots els píxels de la imatge i al trobar dos píxels veïns que tenen identificacions diferents (i no son de fons) és marcarà el valor de la posició actual com a valor a substituir per el de la veïna. També es guarda el valor final com a identificador del TAG quadrat

5.5.1. Pseudocodi

funció (IM_Out,Qua,error)=AlgEtiquetatge2(IM_In,DimIM,k)
Sortides
IM_Out Imatge de sortida
Qua Vector que indica quina posició ocupa el quadrat en el vector que l'identifica
error Escalar, indica si s'ha produït un error o no.
Entrades
IM_In Imatge d'entrada
DimIM Vector amb les dimensions de IM_In
k numero de identificacions de figures diferents

```
Crear Qua amb la mida que indiqui k
Per tot x en IM_In
        Per tot x en IM_In
                Si valor IM_In(posició) és diferent a 1
                        Si veins tenen valor diferent al valor IM_In(posició) i a 1
                                valorEliminar= valor IM_In(posició)
                                valorSubstitut= valor vei
                                Qua(valor  IM_In(posició))=1
                                Per tot x1 en IM_In
                                        Per tot y1 en IM_In
                                                Si valor IM_In(posició1) és igual valorEliminar
                                                        IM_In(posició1) és valorSubstitut
                                                Fi Si
                                        Fi Per
                                Fi Per
                        Fi Si
                Fi Si
        Fi Per
Fi Per
```

5.6. **AlgCercles**

La funció AlgCercles té com a finalitat determinar quines de les siluetes que apareixen en la IM_In passaran a ser considerades TAG.

Per fer això s'utilitzarà la relació entre el perímetre i l'àrea que es detalla en apartat 5.6.1,

Aquesta funció requereix la imatge actual, les seves dimensions, un vector dels elements que la formen i un altre amb l'àrea d'aquests. Com a resultat retornarà les identificacions dels TAG a Elements_Out, el vector Cercle que portarà un 1 en cada posició on tinguem un TAG i Qua_ref, que com en el cas anterior portarà un 1 on tinguem un quadrat mantenint l'ordre de Elements_In, i finalment l'Area_Cercle que tindrà un vector amb les àrees dels TAG.

El primer que es realitza és el càlcul del perímetre. Per calcular-lo repassa la imatge píxel a píxel i comprova, quan es troba amb un píxel que forma part d'una silueta, si algun dels seus veïns directes (en quatre direccions) està identificat com a fons. En cas de ser així incrementarà el valor de la posició del vector perímetre que ocupa la mateixa que en la posició la identificació a Elements_In.

S'aplica la relació àrea perímetre ja mencionada i es guarden els resultats de cada silueta a Cerclecitat, posteriorment filtra segons la seva proximitat al valor de 1. Els tres cercles tindran un valor molt pròxim a 1 i el quadrat rondarà el 0.9, aquesta diferencia permet identificar-los.

Previ a la finalització, la funció realitza una comprovació de que arribats a aquest punt es tinguin 4 elements, en cas contrari s'executa una subrutina que cancel·la la execució del programa i mostra un missatge en un popup que informa del tipus d'error.

5.6.1. Explicació matemàtica

La formula de l'àrea i el perímetre del cercle son les següents:

$$Area\ cercle = \pi \times Radi^2 \quad \text{(Eq. 5.1)}$$

$$Perimetre\ cercle = \pi \times 2 \times Radi \quad \text{(Eq. 5.2)}$$

Per tant si aïllem el radi i el dividim un per l'altre en el cas del cercle tindrem que obtenir un 1 (degut a la discretització el valor no serà exacte) quedant així:

$$Radi = \sqrt{\frac{Area\ cercle}{\pi}} \quad \text{(Eq. 5.3)}$$

$$Radi = \frac{Perimetre\ cercle}{\pi \times 2} \quad \text{(Eq. 5.4)}$$

$$Cerclecitat = \frac{\sqrt{\frac{Area\ cercle}{\pi}}}{\frac{Perimetre\ cercle}{\pi \times 2}} \quad \text{(Eq. 5.5)}$$

5.6.2. Tipus d'error

En aquesta funció és possible que es produeixi l'error de codi 1. Aquest error es produeix quan la rutina és incapaç d'identificar quatre TAGs,i si el codi continués es podrien produir diversos errors:

- Si el numero de TAG fos superior a 4 el codi no detectaria cap error però possiblement es produiria un error en la selecció dels TAGs reals.

- En cas de no activar-se l'error per un numero de TAGs inferior a 4 el programa en produiria un, però la informació no seria suficient com per a que un usuari sense experiència determinés la causa de forma ràpida.

Per solucionar aquest tipusd'error el primer que s'ha de revisar és la imatge capturada. És possible que en la imatge original no apareguin completament els 4 TAGs o que després de la correcció d'inclinació de la càmera respecte l'horitzó algun d'aquests hagi desaparegut.

En el cas de que tots 4 TAGs quedin a la vista podem considerar que es tracta d'un error provocat per l'entorn, per tal d'evitar que no es torni a produir s'haurà de comprovar si després de la funció TrobarTagColor els TAG queden ben definits o no mentre s'identifiquen possibles superfícies que puguin ser erròniament interpretades com a TAG. La il·luminació pot afectar a cóm la càmera recull els colors, com ja s'ha mencionat anteriorment, i es per això que sota certes condicions hauran de variar els valors que estableixen els límits del rang RGB en que considerem que els TAG es diferencien de la resta.

5.6.3. Pseudocodi

Funció(Cercle,Quadrat,Id_Tag_Out)=AlgCercle(IM_In,DimIM,Id_Sil,Area,Qua)

Sortides:
Cercle Vector binari que indica quines siluetes son TAG i quines no
Quadrat Vector binari que indica quina silueta és el TAG quadrat i quines no
Id_Tag_Out Vector amb les identificacions dels TAG
Entrades:
IM_In Imatge d'entrada
DimIm vector amb dimensions imatge d'entrada
Id_Sil Vector amb les identificacionsde les siluetes, possibles TAG
Area Vector amb les àrees de totes les siluetes
Qua Vector amb la id del TAG quadrat

```
Per cada X de IM_In
    Per cada Y de IM_In
        SiValor posició(X,Y) == Id_Sil
            incrementem 1 a Perímetre(valor posició(X,Y) == Id_Sil)
        Fi Si
    Fi Per
Fi Per
Cercliciat=valors equació relacióArea i perímetre(Eq. 5.5)
Per cada TAG
    SiCerclicitat<0.84 o Cerclicitat>1.2
        Cercle = 0
    Sinó
        Cercle =1
    Fi Sinó
Fi Per
Determinació TAG Quadrat
Id_Tag_Out= Id_Sil que siguin Cercle
```

5.7. **CalculCentre**

Aquesta funció serà cridada més d'un cop, la seva funció és la de identificar quin píxel correspon al centre de cada silueta. Per realitzar-hosón necessaris: la imatge on es troben les siluetes IM_In, el vector amb les siluetes que hi apareixen ID_Tag i les mides de la imatge.

El procés es basarà en el càlcul del centre de gravetat.Tot i que les figures a estudi tenen una densitat homogènia la inclinació generarà un petit error en l'eix Y, que en ser un error molt petit es considerarà menyspreable, i es farà de la següent forma:

El primer que es farà és la suma de les posicions de cada píxel en cada eix, i quedarà guardat a PosX i PosY, al mateix temps també es calcularà l'àrea. Aquests tres vectors mantindran l'ordre de ID_Tag, fet que permetrà realitzar càlculs de forma mes senzilla.Un cop s'ha finalitzat el repàs de tota la

imatge es dividirà cada valor dels vectors de posició per la seva area i es tornarà a guardar a PosX i PosY, ara aquests dos vectors contenen el centre de les siluetes.

5.7.1. Explicació matemàtica

$$\frac{1}{M} \times \int r \times \rho \qquad \text{(Eq. 5.6)}$$

On M és la massa total, r la posició i ρ la massa de la porció que representa la posició r. Però si considerem ρ una constant i tenim en compte que l'objecte d'estudi estàdiscretitzat la equació queda simplificada de la següent forma. Per a X(Eq. 5.7) i per a les Y(Eq. 5.8)

$$\frac{1}{A} \times \sum_{i=xf}^{x} \sum_{i=yf}^{y} x \qquad \text{(Eq. 5.7)}$$

$$\frac{1}{A} \times \sum_{i=xf}^{x} \sum_{i=yf}^{y} y \qquad \text{(Eq. 5.8)}$$

5.8. **Eix**

La funció eix s'utilitza per determinar a quina inclinació es troben els TAGs respecte la càmera.

Per realitzar això la funció necessita: la imatge, un vector amb les seves mides, les identificacions dels TAGs i les seves posicions.

Els TAG circular tenen la propietat que la seva inclinació es pot calcular al comparar el seu radi màxim amb el seu radi mínim sense importar des d'on se'ls observi com es pot veure a la Figura 5.10. Per trobar els radis es recorrerà IM_In i cada cop que es troba un píxel s'observa als píxels veïns per saber si és de perímetre ,en els casos que ho sigui es calcularà la distancia fins al seu centroide i es compararà amb els màxims i mínims registrats. Si el nou valor és més gran que el màxim o més petit que el mínim passarà a ser el nou valor.

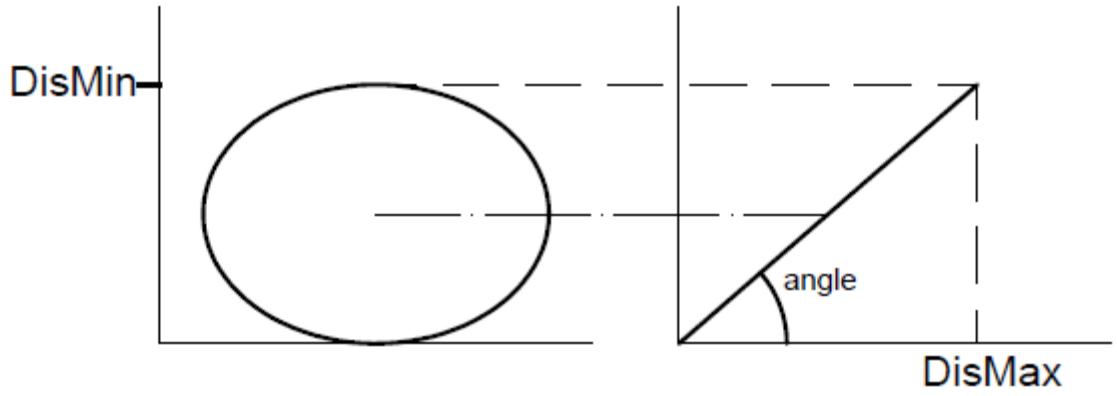

Figura 5.10. Representació de la relació entre els radis màxim i mínim i la inclinació del cercle

Un cop finalitzat el repàs de la imatge amb la relació trigonomètrica s'obté la inclinació d'aquest amb l'equació següent.

$$\cos^{-1}\frac{Radi\ minim}{Radi\ maxim} \qquad \text{(Eq. 5.9)}$$

5.8.1. Pseudocodi

Funció (teta) =AlgEix (IM_In,DimIM,Id_Tag,PosTag)
Sortida:
teta conjunt d'angles que aproximen angle entre el plano i el pla real
Entrada:
IM_In Imatge d'entrada
DimIM Dimensions IM_In
Id_Tag Llista identificacions de TAGs
PosTag Coordenades X Y posició del centre dels TAGs
Per tot X
Per tot Y
Si valor posició actual és diferent a 1
Si valor posició superior o inferior o esquerra o dreta és igual a 1
dist= distancià respecte posició TAG amb mateix valor
Sidist>disMax
disMax=dist
Sinó Sidist<disMin
disMin=dist
Fi Si
Fi Si
Fi Si
Fi Per
Fi Per
teta=arccoseno (disMin/disMax)(Eq. 5.9)

5.9. **AlgCantonades**

La funció AlgCantonades identifica i assigna quin dels TAGs es el més pròxim a cada cantonada. En un primer cas considera que cada TAG queda com el més pròxim a una cantonada, així que identifica quin TAG queda més pròxim a cada cantonada,consultant de forma individual. Aquest procés es realitza ambdosbucles que repeteixen la comparació per cada cantonada i per cada TAG on comparen la distancia entre cantonada i TAG amb la distancia mínima actual.

Al finalitzar aquest bucle comprova si hi ha algun TAG repetit en el vector Punter_Proxim, ho realitza incrementant el valor de la posició homologa en un vector Auxiliar (el vector auxiliar té la mateixa mida que el de TAG i, per exemple en el cas de que Punter_Proxim tingui el TAG numero 3 assignat a més d'una posició, el valor de la posició 3 del vector Auxiliar serà superior a 1). Un cop repassats els quatre punts es revisa si n'hi ha algun que tingui més d'una assignació.En cas de ser així es realitza un bucle on s'estudia la distancia mínima en tot el conjunt.

Aquesta correcció s'ha dissenyat pel cas en el que dos vectors tinguin el mateix valor, i tot i que és cert que en algunes circumstancies es pot donar el cas de que un sol TAG ocupi tres posicions s'ha desestimat realitzar un algoritme per a que ho solucioni ja que nomes es donarà quan l'àrea de treball representi una quarta part de la imatge o menys, i això ja generaria altres problemes que no permetrien executar el codi correctament.

5.9.1. Pseudocodi

```
funcio (Punter_Proxim,Id_Proxim)=AlgCantonades(Id_Tag, Pos_Tag,  DimIM)
Sortides
Punter_Proxim  Vector que recull quin cantonada li toca cada TAG
        "superior esquerra 1, superior dreta 2, inferior esquerra 3, inferior dreta 4"
Id_Proxim      Vector amb els números que identifiquen els TAGs amb cada cantonada
Entrades
Id             Vector amb els TAGs en ordre ordinal
Pos_Tag        Vector amb les posicions dels TAG
DimIM          Dimensions imatge captada
        "En aquesta funció no s'utilitza cap  imatge però si les mides d'una"
Cantonades = Valors extrems de l'imatge segons DimIM
Per cada Cantonada
        Per cada TAG
                Distancia=|Cantonada-Pos_Tag|
                Si Distancia <DisMinActual
                        DisMinActual = Distancia
                        Punter_Proxim = Cantonades
                        Id_Proxim=Id_TAG
                Fi Si
        Fi Per
Fi Per
SiPunter_Proxim te valors repetits
        Matriu_Distancies= Matriu amb totes les distancies entre els TAGs i les Cantonades
        Per primera fila Matriu_Distancies
                Per segona fila Matriu_Distancies
                        Per tercera fila Matriu_Distancies
                                Per quarta fila Matriu_Distancies
                                        Si posicions seleccionades en els bucles son diferents
                                                Distancia_Conjunta=Suma de distancies
                                                Si Distancia_Conjunta<Distancia_Actual
                                                        Distancia_Actual=Distancia_Conjunta
                                                        Id_Proxim=Id_TAG
                                                        Punter_Proxim = Cantonades
                                                Fi Si
                                        Fi Si
                                Fi Per
                        Fi Per
                Fi Per
        Fi Per
Fi Si
```

5.10. **AlgFuga**

La funció AlgFuga calculaamb les dades del posicionament dels TAGs les coordenadesdel punt de fuga, que s'anomenaran X_Fuga i Y_fuga. Correspon al pas 2 de l'apartat 4.2.2.

Per al càlcul de la X_Fuga, al tractar-se d'una càmera, assignarem el seu valor a aproximadament la meitat de l'amplada de la imatge. Per al càlcul del valor d'Y_Fuga es buscarà l'altura d'intersecció de les rectes que formen els TAG assignats a les cantonades esquerra i dreta.

Aquest punt es trobarà igualant les equacions de les rectes que formen i després aïllant el valor en l'eix Y.

5.10.1. Explicació matemàtica

Amb l'equació de la recta(Eq. 5.10) amb els valors de p (pendent) i Y_0 (Y en X =0) per els dos grups de TAGon p s'obté amb (Eq. 5.11) i Y_0 amb (Eq. 5.12) per calcular els de l'altre grup només s'han d'intercanviar els valors per els seus equivalents.

$$Y_{Fuga} = p \times X_{Fuga} + Y_0 \quad \text{(Eq. 5.10)}$$

$$p = \frac{Y_1 - Y_3}{X_1 - X_3} \quad \text{(Eq. 5.11)}$$

$$Y_0 = Y_1 - p \times X_1 \quad \text{(Eq. 5.12)}$$

S'igualen les dos(Eq. 5.13), agrupem X_Fuga(Eq. 5.14) i finalment aïllem X_Fuga(Eq. 5.15).

$$X_{Fuga} \times p + Y_0 = X_{Fuga} \times p' + Y_0{}' \quad \text{(Eq. 5.13)}$$

$$X_{Fuga} \times (p - p') = Y_0{}' - Y_0 \quad \text{(Eq. 5.14)}$$

$$X_{Fuga} = \frac{(Y_0{}' - Y_0)}{(p - p')} \quad \text{(Eq. 5.15)}$$

5.11. **AbatirPlano**

Aquesta funció projecta els TAG al pla que formen les cares superiors dels objectes a identificar. Per realitzar-ho fa servir la imatge IM_In i la seva mida, els identificadors dels TAG , la pendent del pla final i la Y_fuga.

S'utilitzarà Y_fuga ja que tots els plans coincideixen en aquesta altura.

Per evitar que quedin píxels intermedis indefinits com es veu a la Figura 5.11 s'evitarà utilitzar una interpolació directa i es realitzarà amb una indirecta.

Figura 5.11. Representació d'una interpolació directe amb píxels no assignats. [Font: Apunts ARI-EIA EEBE]

Aquesta operació consisteix en recorre la imatge final consultant quin píxel l'hi correspon de la inicial.

La equació que calcularà quina posició de la imatge inicial s'ha de consultar és la següent:

$$k = \frac{i - Y_{Fuga} \times Pendent}{1 - Pendent} \qquad \text{(Eq. 5.16)}$$

On k és la fila de la posició de la imatge final i la posició actual és i.

5.11.1. Explicació matemàtica

Per entendre la equació (Eq. 5.16) s'explicarà , d'entrada,el mètode directe en comptes de l'indirecte. L'equació següent (Eq. 5.17) que correspon a un canvi de posició directe , es pot dividir en dos parts.

La primera "k" representa la posició actual i la segona "(K-Y_Fuga)xPendent" representa la modificació d'aquesta posició, concretament "k-Y_Fuga" indica la distancià del punt actual fins el punt de fuga.

$$i = k - \left(k - Y_{Fuga}\right) \times Pendent \quad \text{(Eq. 5.17)}$$

Un cop vist de forma directa, només cal aïllar la k per tenir la funció en versió indirecta:

$$i - Y_{Fuga} \times Pendent = k - k \times Pendent \quad \text{(Eq. 5.18)}$$

$$i - Y_{Fuga} \times Pendent = k \times (1 - Pendent) \quad \text{(Eq. 5.19)}$$

5.11.2. Pseudocodi

funció (IM_Out)=AbatirPlano(IM_In,ID_Tag,DimIM,Pendent,Y_fuga)
sortides:
IM_Out Imatge de sortida, amb les posicions dels TAGs en el mateix pla que les figures
entrades:
IM_In Imatge d'entrada
ID_Tag Vector amb els valors que identifiquen els TAG
DimIM Vector amb les mides de l'imatge
Pendent Variable que relaciona la desviació en l'eix Y segons l'altura i la proximitat a Y_fuga
Y_fuga Coordenada en l'eix Y on els planós interseccionen.

```
IM_Out = IM_In
Per cada X de IM_In
        Per cada Y de IM_In
                Si valor posició correspon a algun valor de ID_Tag
                        IM_Out(posició)=valor de fons
                Sinó
                        IM_Out(posició)=IM_In(Posició)
                Fi Si
        Fi Per
Fi Per
"Un cop eliminat els TAGs de IM_Out els col·locarà correctament"
Per cada Y de IM_In
        k=Calcula nova posició de les Y amb (Eq. 5.16)
        Per cada X de IM_In
                Si Valor IM_In(posició) correspon a algun valor de ID_TAG
                        IM_Out(X,k)=IM_In(posició)
                Fi Si
        Fi Per
Fi Per
```

5.12. **AlgBase**

Aquesta funció busca el límit inferior on la informació deixa de ser útil. Es considera inútil tot allò que quedaper sota del píxel inferior dels TAG elevats, per tant es repassa la imatge buscant el punt inferior i es deixa un petit marge d'error en els casos possibles per si alguna peça esta en el marge.

5.12.1. Pseudocodi

funció (Y_base)=AlgBase(IM_In, ID_Tag, DimIM)
Sortides:
Y_base Posició límit inferior on la informació deixa de ser útil
Entrades
IM_In Imatge d'entrada
ID_Tag Vector amb les identificacions dels TAG
DimIM Vector amb els mides de IM_In
Per cada X en IM_In
Per cada Y en IM_In
Si valor IM_In(Posició) correspon a ID_Tag
Si Y_Base> Y
Y_Base = Y
Fi Si
Fi Si
Fi Per
Fi Per

5.13. **AlgRelacioPixelRealitat**

Sempre que es vol calcula una distancia o area utilitzant una imatge és necessari tenir una referència, en el nostre cas la referència serà, en un principi, l'amplada del TAG inferior projectat a Y_Base des de X_Fuga. Això és possible ja que aquesta amplada és la que tindrà en la imatge final i es podrà calcular tot comparant aquesta mida amb la resta. Tots els TAG haurien de tenir la mateixa amplada al ser projectats, però la discretització provoca petites variacions.Com més gran es la figura menor percentatge representa cada píxel i per tant menor serà l'error màxim.

El primer que realitzarà el codi és buscar si podem diferenciar els cercles del quadrat ja que aquest no s'ha d'utilitzar degut a que la seva projecció varia depenent de la orientació com es pot veure a la Figura 5.12, si no es així es mostrarà un missatge en una finestra emergent.

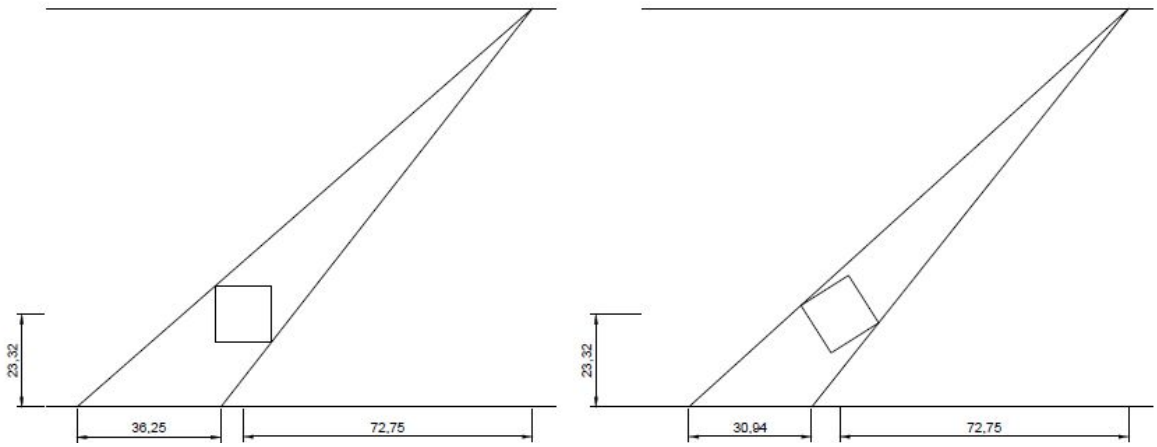

Figura 5.12. Comparació distancia projectada segons inclinació.

Un cop tenim el millor TAG possible es recorre la imatge buscant la seva posició i es guarda el valor de la projecció (segons (Eq. 5.20))més gran i més petit, la diferencia entre aquest dos valors serà la RelacióPixelRealitat i equivaldrà a 2cm en la realitat.

$$X_{base} = \Big((Y_{base} - Y) \times (X_{fuga} - X)\Big) \Big/ (Y_{fuga} - Y) + X \qquad \text{(Eq. 5.20)}$$

5.13.1. Pseudocodi

Funció (RelacioPixelRealitat)= AlgRelacióPixelRealitat(IM_In, DimIM, Id_total, Id_Tag, Cercle, Quadrat, X_Fuga, Y_Fuga, Y_base, Id_Proxim, Punter_Proxim, PosY)

Sortides

RelacioPixelRealitat	Valor escalar que indica a quant equival un píxel en la realitat

Entrades

IM_In	Imatge d'entrada
DimIM	Vector amb les dimensions de IM_In
Id_total	Vector amb identificació de tots els elements
Id_Tag	Vector amb identificació dels TAG
Cercle	Vector que identifica les TAG (segons identificacions de Id_total)
Quadrat	Vector que identifica el quadrat (segons identificacions de Id_total)
X_Fuga	Posició en eix X del punt de fuga
Y_Fuga	Posició en eix Y del punt de fuga
Y_Base	Límit inferior de sona interessant
Id_Proxim	Vector amb identificador de TAG segons proximitat a les cantonades
Punter_Proxim	Vector amb l'ordre de les cantonades segons proximitat amb el TAG
PosY	Vector amb les posicions dels TAG (ja adaptades apartat 5.11)

```
Comprovació correcte identificació quadrat.
Selecció TAG més pròxim a Y_Base que no es quadrat
Per cada X en IM_In
        Per cada Y en IM_In
                Si valor IM_In(posició) == TAG seleccionat
                        K=Projectar posició a Y_base(Eq. 5.20)
                        Si K<esquerra
                                esquerra=K
                        Sino Si K> dreta
                                dreta= K
                        Fi Si
                Fi Si
        Fi Per
Fi Per
RelacioPixelRealitat =dreta-esquerra
```

5.14. **AlgOrientacioA**

La funcióAlgOrientacioA retorna la distancia d (Figura 4.8) que s'utilitza per realitzar la transformació de la imatge i també desde quin canto estem observant la matriu dels TAG.

Ho realitzacomprovant en quina posició està el quadrat. Per identificar el cantó necessita la identificació de quin TAG queda en cada cantonada, la llista d'elements amb la que s'ha realitzat la llista de TAG i el vector on es determina si una silueta és o no el TAG quadrat. Un cop identificat per

determinar la d necessitarà les mesures de la zona de treball en relació a la mida dels TAG, la variable RelacioPixelRealitat i la inclinació de la matriu respecte el pla .

5.14.1. Pseudocodi

Funció(D,direccio)= AlgOrientacioA(Id_Proxim, Id_total, Quadrat, Amplada, Profunditat, X_fuga, Y_fuga, Y_base, Pos, Punter_Proxim)

Sortides

D	Distancia entre TAG eix Y Figura 4.8
direccio	Indicador del punt de vista de la càmera

Entrades

Id_Proxim	Vector amb la relació entre cantonades
Id_total	Vector amb identificació de tots els elements
Quadrat	Vector que identifica el quadrat (segons identificacions de Id_total)
Amplada	Constant que relaciona amplada zona amb amplada TAG
Profunditat	Constant que relaciona profunditat zona amb amplada TAG
X_Fuga	Posició en eix X del punt de fuga
Y_Fuga	Posició en eix Y del punt de fuga
Y_Base	Límit inferior de sona interessant
Punter_Proxim	Vector amb l'ordre de les cantonades segons proximitat amb el TAG
PosY	Vector amb les posicions dels TAG (ja adaptades apartat 5.11)

direccio=Relaciona Id_Proxim, Quadrat i Id_Total
"Si Id quadrat esta en la posició de Id_Proxim es :1 esquerra, 2 davant, 3 darrere, 4 dreta"
Xa=Projeccio TAG 3 a Y_base
Xb=Projeccio TAG 4 a Y_base
Sidireccio és esquerra o dreta
    D=(Xb-Xa)(Amplada/Profunditat)
Sinó
    D=(Xb-Xa)(Profunditat/Amplada)
Fi Si

5.15. **FuncioAltura**

En aquesta funció es determinaran les variablesX_altura i X_alturae, que ens permetran realitzar la transformació, ja que com es pot observar en l'apartat 4.2.son les encarregades de definir la profunditat.

Per poder aplicar l'algoritme és necessari que els dos TAG a utilitzar com a punts de referènciaes trobin en el mateix canto de la columna de X_fuga. Un cop es tenen els dos TAG que s'utilitzaran es

projecta des de X_Fuga a Y_Base passant per els centres dels TAG i se'ls anomena a i b, amb aquest i la resta de punts ja obtinguts es calcula la següent equació

$$X_{altura} = \frac{f1 + f2 - f3}{f4} \quad \text{(Eq. 5.21)}$$

El mateix càlcul serveix per X_alturae. Al calcular-ne un, l'altre s'obté per simetria a X_fuga.

$$f1 = (D + X_a - X_b - a + b) \times \left(Y_{fuga} - Y_a\right) \times \left(Y_{fuga} - Y_b\right) \quad \text{(Eq. 5.22)}$$

$$f2 = X_b \times (Y_{base} - Y_b) \times \left(Y_{fuga} - Y_a\right) \quad \text{(Eq. 5.23)}$$

$$f3 = X_a \times (Y_{base} - Y_a) \times \left(Y_{fuga} - Y_b\right) \quad \text{(Eq. 5.24)}$$

$$f4 = (Y_{base} - Y_b) \times \left(Y_{fuga} - Y_a\right) - (Y_{base} - Y_a) \times \left(Y_{fuga} - Y_b\right) \quad \text{(Eq. 5.25)}$$

5.15.1. Explicació matemàtica

L'equació principal de l'apartat anterior (Eq. 5.21) s'obté en utilitzar la relació de D (Eq. 5.26). Amb l'objectiu d'aïllar X_{altura} i tenint en compte que també es desconeixen els valors δa i δb, substituïm els valors de la equació de D per les equacions (Eq. 5.29) i (Eq. 5.30). Un cop substituïdes ja es pot aïllar.

$$D = |\delta_b - b| - |\delta_a - a| \quad \text{(Eq. 5.26)}$$

$$a = \Big(\big(Y_{base} - Y_{fuga}\big) \times \big(X_a - X_{fuga}\big)\Big) \Big/ \big(Y_a - Y_{fuga}\big) + X_{fuga} \quad \text{(Eq. 5.27)}$$

$$b = \Big(\big(Y_{base} - Y_{fuga}\big) \times \big(X_b - X_{fuga}\big)\Big) \Big/ \big(Y_b - Y_{fuga}\big) + X_{fuga} \quad \text{(Eq. 5.28)}$$

$$\delta_a = \Big(\big(Y_{base} - Y_{fuga}\big) \times \big(X_a - X_{altura}\big)\Big) \Big/ \big(Y_a - Y_{fuga}\big) + X_{altura} \quad \text{(Eq. 5.29)}$$

$$\delta_b = \Big(\big(Y_{base} - Y_{fuga}\big) \times \big(X_b - X_{altura}\big)\Big) \Big/ \big(Y_b - Y_{fuga}\big) + X_{altura} \quad \text{(Eq. 5.30)}$$

Per una millor comprensió veure Figura 4.7 i Figura 4.8 on es pot veure a que fa referència cada símbol.

5.16. **AlgLimitZonaTreball**

Com ja s'ha realitzat en la funció AlgBase (apartat 5.12) per a calcular el límit base, s'han d'obtenir els altres tres límits de la zona útil. Però a diferencia de Y_base aquests límits no es calcularan per a la imatge inicial sinó que es realitzaran per a la imatge final.

S'utilitzarà una projecció simple de X_fuga a punts de l'altura de Y_base passant per tots els píxels que corresponen a TAGs i es guardarà la columna de la projecció més allunyada per cada cantó. Amb aquesta tècnica determinarem els límits d'esquerra i dreta.

Per determinar el límit superior aprofitarem la projecció ja realitzada i es combinarà amb una altre desdeX_altura o X_alturae fins a Y_base passant per cada punt. El valor que es compararà per determinar el punt superior és la diferencia entre aquestes dues projeccions.

5.16.1. Pseudocodi

Funció (AmaxEsquerra, AmaxDreta, superior)= AlgLimitZonaTreball (IM_In, Id_Tag, DimIM, X_Fuga, Y_fuga, Y_base, X_altura, X_alturae)

Sortides	
superior	Límit superior de sona interessant
AmaxEsquerra	Límit esquerra de sona interessant
AmaxDreta	Límit dreta de sona interessant
Entrades	
IM_In	Imatge d'entrada
Id_Tag	Vector amb identificacions dels TAG
DimIM	Vector dimensions IM_In
X_Fuga	Posició en eix X del punt de fuga
Y_Fuga	Posició en eix Y del punt de fuga
Y_Base	Límit inferior de sona interessant
X_altura	Punt de projecció per profunditat esquerra
X_alturae	Punt de projecció per profunditat esquerra

```
Per tot X de IM_In
        Si X< (X/2)
                Altura<-AlturaDreta
        Sinó
                Altura<- AlturaEsquerra
        Fi Sinó
        Per tot Y de IM_In
                A= Valor X de Projecció X_fuga a Y_base passant el punt X,Y corresponent a TAG
                Si A>AmaxEsquerra
                        AmaxEsquerra=A
                Fi Si
                Si A>AmaxDreta
                        AmaxDreta=A
                Fi Si
                B=Projecció Altura a Y_base passant el punt X,Y corresponent a TAG
                Si B-A >Superior
                        Superior =B-A
                Fi Si
                Si A-B>Superior
                        Superior= A-B
                Fi Si
        Fi Per
Fi Per
```

5.17. **AlgTransformacio**

La funció AlgTransformacio executarà la seqüència que permetrà crear la nova imatge on es representarà la vista del pla a estudiar com si fos en vista zenital. Per realitzar això seran necessaris tots els punts citats en l'apartat 4.2, la imatge d'entrada IM_In i els límits que establiran des d'on i fins a on s'ha de transformar la imatge.

El primer que es calcularà serà la posició relativa entre els punts d'entrada que corresponen a la imatge inicial i els que corresponen a la imatge final, per exemple, el punt X_fuga representa el punt mig de la imatge, per tant si la imatge canvia de mida aquest també canviarà la coordenada.

Aquesta funció es farà servir per una imatge en escala de grisos (que nomes té una capa de profunditat) i per una en format RGB (que té tres capes de profunditat), per això es consultarà quin tipus d'imatge d'entrada té.

La transformació es realitzarà píxel a píxel d'esquerra a dreta i de baix a d'alt, i per el mateix motiu que en l'apartat 5.11 es realitzarà amb una interpolació indirecta però en el cas de la imatge RGB es realitzarà utilitzant una aproximació quadràtica als píxels del voltant del punt seleccionat.

Els límits s'han establert amb la imatge original però al realitzar la projecció és possible que en alguns extrems l'algoritme intenti consultar alguns punts fora dels límits de la imatge original. Per evitar errors es comprovarà prèviament si els punts que intenta consultar estan dintre del límit.

5.17.1. Pseudocodi

funcio (IM_Out)=AlgTransformacio(IM_In,X_altura, X_alturae, X_fuga, Y_fuga, Y_base, superior, inici, final)

Sortides	
IM_Out	Imatge de sortida
Entrades	
IM_In	Imatge d'entrada
X_altura	Punt de projecció per profunditat esquerra
X_alturae	Punt de projecció per profunditat esquerra
X_Fuga	Posició en eix X del punt de fuga
Y_Fuga	Posició en eix Y del punt de fuga
Y_Base	Límit inferior de sona interessant
superior	Límit superior de sona interessant
inici	Límit esquerra de sona interessant
final	Límit dreta de sona interessant

Adaptació valors coordenades trobades a la nova mida d'imatge.

```
Per tot X en IM_In
        Per tot Y en IM_In
                Si X< (X/2)
                        Altura=AlturaDreta
                Sinó
                        Altura=AlturaEsquerra
                Fi Sinó
                X,Y=valor imatge original d'intersecció de les línees(veure Figura 4.8)
        Fi Per
Fi Per
```

5.18. **FuncioTrobarFigura**

La funció TrobarFigura, com la FunciónTrobarTag (apartat 5.2)però amb menys restriccions, segmenta la imatge separant la informació útil de la inútil, en aquest cas separa els píxels que per codi de colors RGB poden ser considerats figures de la resta.

Per fer aquesta funció nomes necessita la imatge i les dimensions d'aquesta i torna la imatge ja segmentada. El procés és simple, repassa cada píxel i comprova si compleix amb les relacions entre els valors RGB.

5.18.1. Pseudocodi

```
función (IM_Out)= FuncioTrobarFigures(IM_In,DimIM)
sortides
IM_Out          Imatge segmentada
entrades
IM_In           Imatge d'entrada
DimIM           Vector amb les mides de IM_In
Per tot X de IM_In
        Per tot Y de IM_In
                Si  valor IM_In(posició) és blau
                        IM_Out(posició)=1
                Sinó
                        IM_Out(posició)=0
                Fi Si
        Fi Per
Fi Per
```

5.19. **CalculArea**

En aquesta funció es repeteix el que apareix en la funció Main, entre l'aplicació de la funció AlgElimOmbres i AlgCercles però adequant-ho a un moment en el que ja es té certa informació.

El primer que fa la funció és calcular l'àrea de la imatge d'entrada, elimina les àrees més petites que els TAG i les vint vegades més grans. Finalment retorna només les àrees restants junt amb la llista de les identificacions.

5.20. **FuncioMarges**

La funció FuncioMarges s'aplicarà a la imatge transformada en color i ja etiquetada,repassant tots els punts de la imatge on es trobi un píxel d'una figura i nomes deixarà en el mateix estat els que corresponen al marge de la figura.A la resta els assignarà un 0, d'aquesta manera en futures funcions s'entendrà que tot i que formen part de la figura no s'han d'utilitzar en operacions que busquin informació relacionada amb la forma.

També s'aprofitarà aquest bucle per comptar els píxels que formen part del marge, valor que s'assignarà al perímetre de la figura.

5.20.1. Pseudocodi

```
Funció (IM_Out,Perimetre)=AlgMarges(IM_In,DimIM,Id_Tag)
Sortides
IM_Out          Imatge de sortida
Perimetre       Vector amb els  perímetres de les figures
Entrades
IM_In           Imatge d'entrada, segmentada.
DimIM           Vector amb dimensions imatge d'entrada
Id_Tag          Vector amb identificacions de les figures

Per tot X de IM_In
        Per tot Y de IM_In
                Si valor IM_In(posició) correspon a algun valor de  Id_Tag
                        Si valor IM_In(posicions voltant) igual a valor IM_In(posició)
                                IM_In(posició)=fondo
                        Sinó
                                incrementar perimetre
                        Fi Si
                Fi Si
        Fi Per
Fi Per
```

5.21. **AlgVertex**

Els humans identifiquen fàcilment una figura mitjançant paràmetres com el número de cantons, la relació de mides que tenen, els angles que la formen i la mida. Això és possible gracies a que som capaços d'identificar quina part del que estem veient forma part de la figura i quina no, i podem veure patrons com línees i vèrtex. existeixen varismètodes per a que un ordinador aconsegueixi el mateix, un d'ells és la transformada de Hough. Aquesta transformada permet trobar rectes segons el numero de punts que poden formar part d'una recta a la mateixa distancia del punt origen i amb el mateix angle. Per això sol ser utilitzada en processos de control per visió.

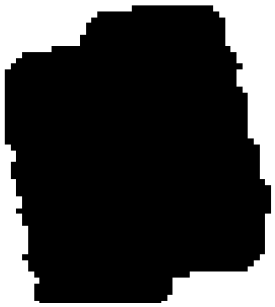

Figura 5.13. Silueta d'un quadrat després de ser tractat amb una sèrie de filtres i transformades.

Al aplicar-la en aquest projecte es trobaria amb figures com la Figura 5.13 on sobre tot en les línees amb pendent pròximes a la horitzontal es poden veure deformacions , això faria que s'haguésd'utilitzar la funció amb una alta tolerància, fet que impediria detectar figures com un hexàgon ja que el detectaria quadrat.

Per aquest motiu s'ha dissenyat un sistema més concret, que pugui estudiar els possibles casos que poden aparèixer en la planta dissenyada.

Les característiques d'àrea o perímetre permeten la diferenciació entre siluetes, sobre tot si poden tenir la mida exacte, però no és el cas. Al tractar-se d'una imatge discretitzada, transformada i filtrada les siluetes han perdut la forma. Per això s'hauràd'utilitzar un mètode mixte.

Aquesta funció realitza la cerca de vèrtexs. Per localitzar-los s'utilitza la imatge procedent de la funció FuncioMarges (apartat 5.20) on es té separat el cos delspíxels que formen el perímetre, i avalua cada punt del marge comparant-lo amb els deu que l'envolten per comprovar si el radi en aquest punt és un màxim local o no. En els casos que així sigui es guardaran les posicions i el valor del radi en la matriu Maxims. Aquesta inspecció es realitzarà en ordre, un cop es trobi per primer cop una silueta es seguirà utilitzant el mateix mètode que la funció AlgTortPapert (apartat 5.3.1) aconseguint així que es puguin comparar les posicions dels valors que es troben a la matriu Maxims.

La matriu Maxims modifica la seva longitud varies vegades cada cop que s'avaluen els punts d'una figura, en cada pas s'incorporaran a cada canto de la matriu una part d'aquesta per tal de facilitar la comparació entre els extrems de la matriu. Després de realitzar cada filtre elimina de la matriu tan els punts desestimats com les columnes que s'havien afegit i es tornarà a redimensionar la matriu per al següent procés.

El següent filtre consta d'una comparació de la distancia entre dos punts i una novena part del perímetre. S'aplica aquest filtre per eliminar, per exemple, múltiples punts d'un mateix vèrtex ja que poden aparèixer degut a la irregularitat. S'ha utilitzat el valor d'una novena part del perímetre ja que

cap de les figures té més de sis cares, però no es pot utilitzar un valor més exacte ja que les figures no son equilàters.

Figura 5.14. Dues siluetes on s'observa la irregularitat en els seus laterals.

Arribats a aquest punt s'han eliminat molts dels màxims locals, però com es pot veure a la Figura 5.14 els cantons més llargs que representen més d'una cinquena part del perímetre de la silueta encara es podrien considerar molts punts com a possibles vèrtex, per això s'aplicarà un filtre que compararà les rectes que forma cada punt amb els punts anteriors, aquest màxims intermedis es troben en una recta i per tant l'angle que formaran serà superior a l'angle que tindria una forma geomètrica.

5.21.1. Pseudocodi

```
Funció (Info)=AlgVertex(IM_In, Id, Pos, Perimetre, DimIM)
Sortida
Info            Matriu de 4x6x numero de capes
Entrades
IM_In           Imatge d'entrada
Id              Vector amb identificador de les figures
Pos             Vector amb les posicions de les  figures en IM_In
Perímetre       Vector amb els perímetres de les figures
DimIM           Vector amb les dimensions de IM_In
Per tot X de IM_In
        Per tot  Y de IM_In
                Per cada Id
                        Si valor posició és igual a Id
                                Mentre posició diferent a posició' (mètode AlgTortPapert)
                                        Maxims és igual a distancia entre posició i Pos
                                Fi Mentre
                                Per cada Maxims
                                        SiMaxim actual >maximsveins (10)
                                                Maxim actual marcat com a màxim local
                                        Sinó
                                                Maxim actual marcat com a no màxim local
                                        Fi Si
                                        Si distancia entre Maxims continus <perímetre/9
                                                Maxim menor marcat com a no màxim local
                                        Fi Si
                                        Si Angle entre Maxims continus <130
                                                Maxim menor marcat com a no màxim local
                                        Fi Si
                                Fi Per
                        Fi Si
                Fi Per
        Fi Per
Fi Per
```

5.22. **AlgIdentificacio**

Amb la informació extreta en les funcions anteriors AlgVertex (apartat 5.21) i CalculArea (apartat5.19) es procedirà a identificar a quin dels valors introduïts a la biblioteca corresponen. Filtra cada figura segons el numero de vèrtexs per identificar el tipus de polígon is'especifica amb consultes de si son o no equilàters o equiangulars, en acabar l'anàlisi sel'hi adjudica un numero que posteriorment a la funció main li permetrà adjudicar-li un nom.

5.22.1. Pseudocodi

funcio (NomFigura)=AlgIdentifiacio (Info_Fig, Area_Fig, Base_Dades, RelacioPixelRealitat)
Sortides
NomFigura Vector que relaciona el nom de les figures amb la posició en el vector que les identifica
Entrades
Info_Fig Matriu amb informació obtinguda en la funció AlgVertex5.21
Area_Fig Vector amb area figures
Base_Dades Matriu amb les constants que defineixen les diferents figures
RelacioPixelRealitat Valor que relaciona la distancia entre pixel i reailtat

```
Per cada Info_Fig
        Si vertex= 6
                NomFigura(Info_Fig)= Hexagon
        Sino Sivertex= 5
                NomFigura(Info_Fig)=Pentagon
        Sino Sivertex= 4
                Si equiangular
                        Siequilater
                                Si Area< Base_dades
                                        NomFigura(Info_Fig)=Quadrat Petit
                                Sino
                                        NomFigura(Info_Fig)=Quadrat Gran
                                Fi Si
                        Sino Si Area<Base_dades
                                NomFigura(Info_Fig)=Rectangle petit
                        Sino
                                NomFigura(Info_Fig)=Rombe
                        Fi Si
                Sino
                        NomFigura(Info_Fig)=Rombe
                Fi Si
        Si vertex = 3 "triangle"
                Siequilàter
                        NomFigura(Info_Fig)="Triangle equilàter"
                Sino
                        NomFigura(Info_Fig)="Triangle escalar"
                Fi Si
        Fi Si
Fi Per
```

5.23. **AlgOrientacioB**

AlgOrientacioBés una funció que calcula la desviació de l'estructura de TAG respecte l'eix de la imatge, es necessària per poder indicar en quina posició real es troba cada figura.Primer es calcula el

posicionament del centroide dels TAG en la imatge final evitant així transformar tota la imatge, fet que requeriria més temps. Un cop tenim els nous punts es calculà l'angle respecte la vertical i es torna com a variable de sortida.

Com s'havia fet en altres funcions, es comprova que els dos TAG a utilitzar estiguin en el mateix canto de X_fuga i s'evitarà utilitzar el cantó que tingui el quadrat ja que és el TAG que perd més informació amb la inclinació de la càmera.

5.24. **RotacioPosicions**

La funció RotacioPosicions es la ultimà funció necessària per designar en quina posició està cada figura. En aquesta funció es rota la posició del centroide de les figures i es calcula la posició del punt respecte el TAG quadrat, posició que s'utilitzaràcom a punt 0,0 sigui quina sigui la orientació.

5.24.1. Explicació matemàtica

Per calcular la posició el codi transformarà les dues posicions en un vector, després traslladarà el seu origen a 0, canviarà la seva orientació a la designada per l'angle calculat en la funció AlgOrientacioB, es tornarà l'origen del vector on ja el tenia i es tornarà a la posició original. Finalment l'altre extrem del vector serà la posició del centroide un cop centrat l'eix de la estructura de TAG i la càmera.

$$PosX_{Final} = \sqrt{PosX_{cor}{}^2 + PosY_{cor}{}^2} \times \sin(-teta + teta2) \quad \text{(Eq. 5.31)}$$

$$PosY_{Final} = \sqrt{PosX_{cor}{}^2 + PosY_{cor}{}^2} \times \cos(-teta + teta2) \quad \text{(Eq. 5.32)}$$

On les posicions PosX_cor i PosY_cor són les distancies a cada eix del centroide, si el TAG sobre el que es rotarà el considerem com a punt origen, teta es l'angle de la funció AlgOrientacioB i teta2 es l'angle original del vector.

Un cop rotades les posicions, calculà la diferencia entre la posició actual i el TAG quadrat quedant ja la distancia que ens interessa en píxels. Finalment nomes cal transformar-ho a cm utilitzant la relacioPixelRealitat.

5.25. **Mostrar Errors**

Aquesta petita funció s'encarrega de mostrar i aturar el programa quan es produeix un error comú, com per exemple els errors produïts per problemes d'il·luminació.

En aturar el programa es mostra perla consola de MATLAB el missatge que indica quin tipus d'error ha provocat la aturada.

5.26. **AlgIdentificarImatge**

Aquesta és una funció de demostració. No s'utilitzaria en un ús normal però resulta molt útil per mostrar els resultats. La finalitat d'aquesta funció és mostrar un quadre de text sota el centre de cada figura, indicar de quina es tracta i de quina posició ocupa en el món real.

6. Calibratge càmera

Les càmeres web, com també altres càmeres sofreixen de distorsió d'imatge. Aquest efecte produeix que per algunes zones un píxel equivalgui a més espai en la realitat que per altres píxels, això provoca una distorsió, com es pot apreciar en la Figura 6.1. on si s'observa el marge esquerra del "taulell d'escacs" de l'imatge A), es pot observar una corba sobre tot en la part inferior d'aquest però que en la imatge B) ,ja corregida, aquesta línea és perfectament recta.

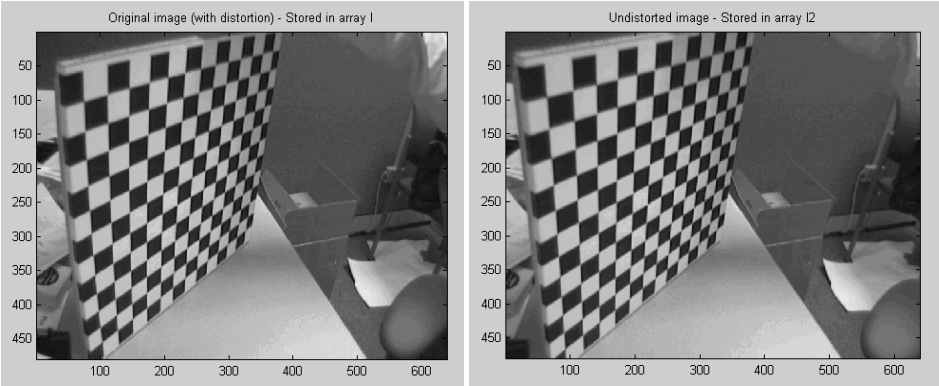

Figura 6.1.A)Imatge original B)Imatge corregida.[6]

Per realitzar aquesta correcció s'ha fet servir la aplicació que incorpora MATLAB, que fa servir una sèrie de paràmetres que podem separar en dos grans grups, paràmetres intrínsecs i paràmetres extrínsecs que es detallaran a continuació.

MATLAB també requereix la introducció de la mida real del patró utilitzat com a matriu (veure apartat 6.3) i reconeix la mida de la càmera.

6.1. Paràmetres intrínsecs

Els paràmetres intrínsecs són tots aquells paràmetres que caracteritzen la càmera.

Distancia focal

La distancia focal o longitud focal és la distancia que hi ha entre la lent i el punt en el que convergeixen dos punts, a priori paral·lels, en el moment de creuar la lent.

MATLAB ho recull com un vector de 2x1 anomenat focal length, la distancià es troba en píxels.

Punts principals

Per parlar dels punts principals és necessari primer determinar què són els plans principals.

Els plans principals són els plans que son creuats a la mateixa distancia del centre del pla que l'objectiu(efecte degut a la refracció). Si la refracció es nul·la el pla podria estar situat en qualsevol punt ja que sempre es compliria[7]. Els punts principals són la posició en la que es creua el pla principal amb el pla de la lent.

MATLAB ho recull en un vector 2x1 anomenat principal point.

Coeficient de distorsió

Determina la distorsió respecte la perfecció entre l'eix X i l'eix Y de la imatge. En cas de que no hi hagués 90˚ entre ells, el valor de coeficient de distorsió aniria disminuint des d'1 fins a 0.

MATLAB ho recull com un escalar i l'anomena skew.

Distorsions

Les distorsions poden seguir patrons, entre els quals MATLAB treballa amb distorsió radial i distorsió tangencial, aquesta segona de forma opcional.

La explicació física d'aquestes distorsions és, en el cas de la radial,la diferencia entre el radi real i el radi ideal que hauria de tenir la lent i, per el que fa a la distorsió tangencial, ens indica que el centre de la lent es troba desviat del punt que la càmera interpreta com a centre de la imatge. Figura 6.2

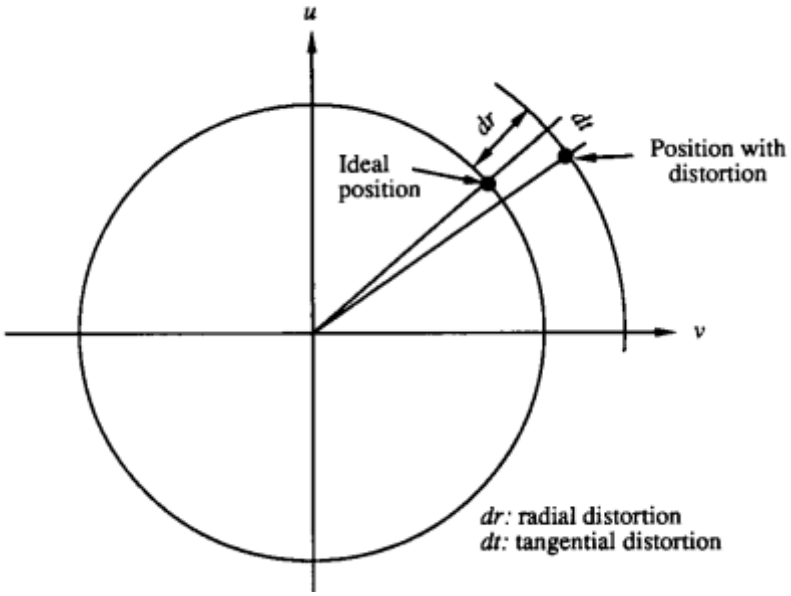

Figura 6.2.**Representació física del motius de la distorsió[8]**

Amb aquests patrons MATLAB aproxima quina és la distorsió real. Amb els codis obtinguts a [9].es crearan mapes per entendre com es veuen afectades les imatges. En la Figura 6.3A) es pot veure com és el mapa de la distorsió radial, al tractar-se d'una càmera amb un objectiu molt petit la distorsió és molt gran, per aquest motiu es crea un patró tant pronunciat. L'imatge B) és el resultat amb uns valors inferiors amb la fi de demostrar millor visualment com treballa.

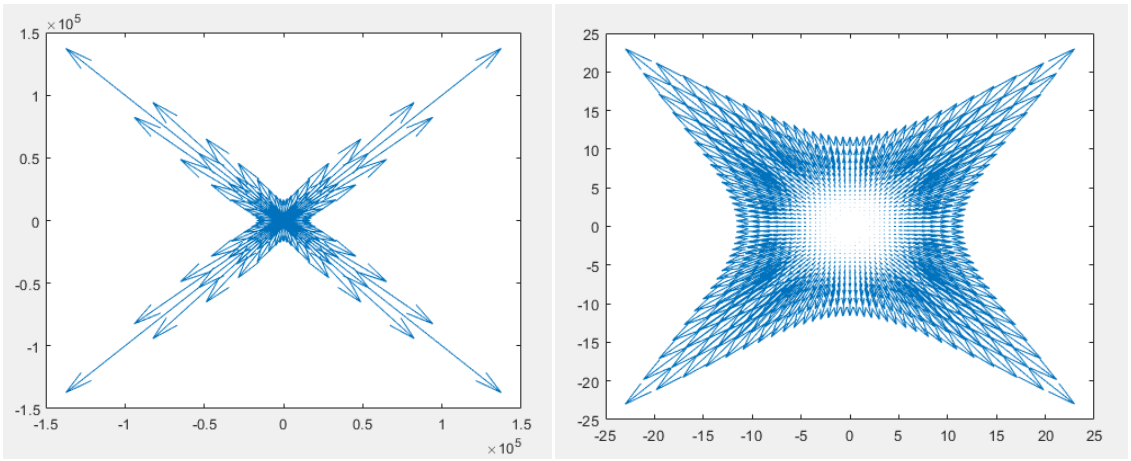

Figura 6.3.A)Valors de la càmera utilitzada B)Valors per a millor visualització

Tot i que en els dos casos exposatsMATLAB utilitzi un vector de 1x3 es podria pensar que es tracta d'una aproximació cubica però la realitat es que el tercer valor és en potencia de 4. L'altre tipus de distorsió com ja s'ha mencionat és la distorsió tangencial. Figura 6.4

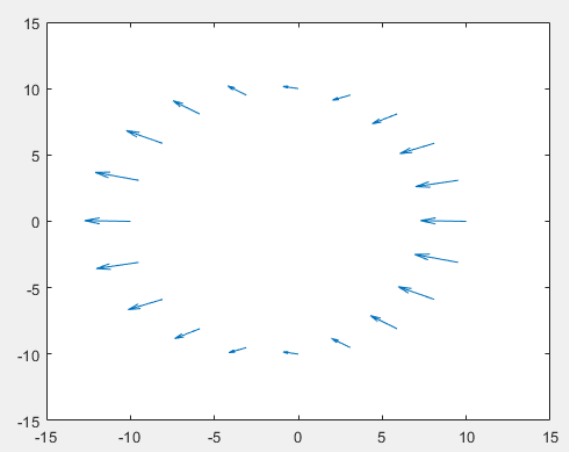

Figura 6.4.**Valors de la càmera utilitzada**

6.2. **Paràmetres extrínsecs**

Els paràmetres extrínsecs són els paràmetres que fan referència al punt 0,0 i al pla de la matriu que s'utilitzarà per realitzar el calibratge (veure apartat 6.3).

De paràmetres extrínsecs n'hi ha dos, de translació i de rotació.

Translació

Distància entre el punt cèntric de captació de la imatge i el punt 0,0 de la matriu que s'utilitza. MATLAB ho recull en un vector de 1x3 en mil·límetres.

Rotació

Indica la rotació del sistema ortogonal del pla de la matriu respecte la posició ortogonal que espera la càmera o també anomenat angles d'Euler, els valors angulars entre un sistema i l'altre. MATLAB també ho recull com un vector de 1x3.

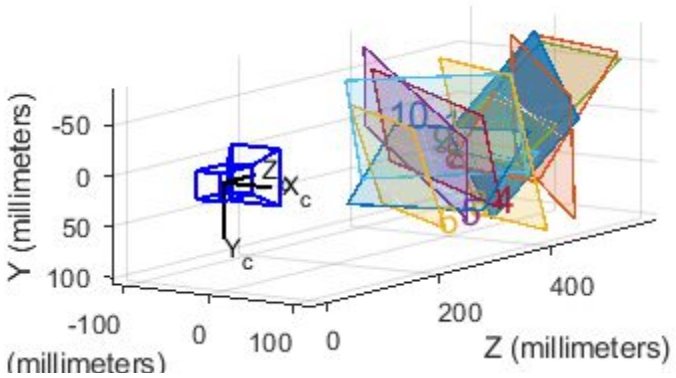

Figura 6.5.**Representació de les dades extrínseques captades per MATLAB amb les imatges proporcionades**

6.3. **Plantilla**

Per la realització del calibratge és necessària una plantilla, en diferents programes s'utilitzen diversos patrons, però tots comparteixen el fet de que son repetitius, clars, i amb una alta definició.

MATLAB utilitza un patró que segueix una estructura de taulell d'escacs on és important que el dibuix no sigui simètric respecte la diagonal, ja que d'aquesta manera no confondrà el patró al posar-lo en una rotació superior a 180˚.

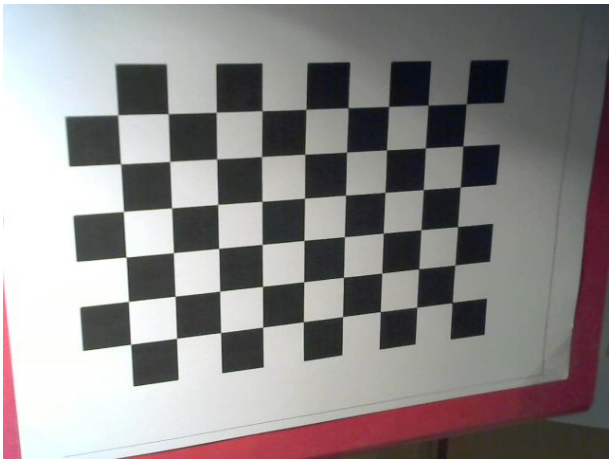

Figura 6.6.**Matriu utilitzada per el calibratge .Correspon al pla nº7 de la** Figura 6.5 **en color cian.**

6.4. **Ús de l'aplicació d'autocalibratge**

Per realitzar l'operació d'autocalibratge en MATLAB 2017 es clicarà a APPS en el menú superior i de les opcions que apareixeran es selecciona CameraCalibrator (veure Figura 6.7)

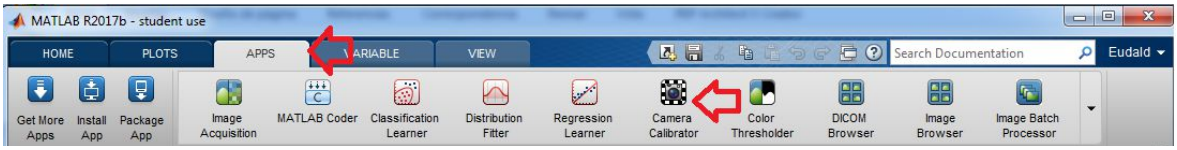

Figura 6.7.**Imatge del menú general de MATLAB.**

S'obrirà una nova finestra on afegirem el conjunt d'imatges preses anteriorment o les realitzarem en el moment desplegant la opció AddImages a la paleta File (veure Figura 6.8). Al carregar les imatges el programa consultarà a l'usuari quina és la mida dels quadrats utilitzats a la matriu (important no confondre les unitats). Al carregar una imatge MATLAB l'analitzarà i trobarà ja els punts vèrtex d'unió entre quadrats i assignarà el punt 0,0. En aquest punt podrem ampliar les imatges per comprovar si ha realitzat correctament aquesta operació ja que si no es així el calibratge serà erroni. Un cop considerem que ja són suficients imatges seleccionarem quin tipus d'aproximació volem en la paleta Options, i finalment clicarem a Calibrate perque realitzi l'autocalibració.

Figura 6.8.**Imatge del menú CameraCalibrator de MATLAB.**

7. ANÀLISIS DE L'IMPACTE AMBIENTAL

Aquest treball no genera residus mediambientals a excepció del la substitució de la càmera web en cas d'espatllar-se.

Les figures són reutilitzades del laboratori de robòtica industrial i visió per computació i en acabar el període de presentacions seran tornades per el seu ús en futures promocions.

Per altre banda la tecnologia de control per visió pot ser utilitzada per a feines de reciclatge on actualment es troben treballadors fent feines de filtratge de productes reciclables dels que no ho són, i per tant parts d'aquest codi podrien ser utilitzades en la classificació i ajudar així a pal·liar la petjada humana.

8. Resultats

En aquest capítol és mostrarà el procés i els resultats de la validació sobre el terreny (comprovació de la correspondència entre les dades obtingudes per la aplicació amb les reals). El primer que es comprova és l'error en l'àrea de les peces i després l'error en el posicionament.

Les imatges per realitzar els apartats 8.2 i **¡Error! No se encuentra el origen de la referencia.** s'ha realitzat des de la cantonada superior esquerra.

8.1. Area real

El càlcul de l'àrea s'ha realitzat utilitzant un peu de rei per obtenir les mides i les formules geomètriques per calcular l'àrea.

8.1.1. Hexàgon

On D és el diàmetre del cercle intrínsec (dues vegades Ap).

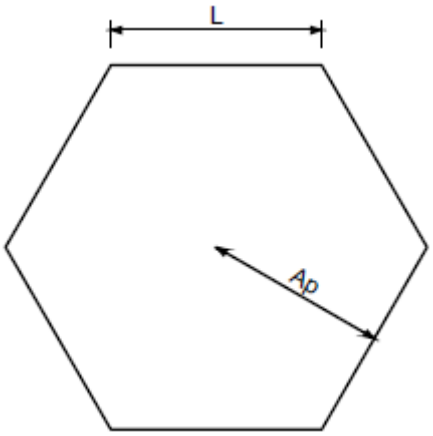

Figura 8.1.**Representació de les mesures realitzades en els hexàgon.**

$$Àrea = L \times Ap \times 3 \qquad \text{(Eq. 8.1)}$$

D	L	Ap	Àrea
39	22	19,5	1287
38,4	21	19,2	1209,6
38	22,4	19	1276,8
Mitjana			1257,8

Taula 1Mesures hexàgon

8.1.2. Pentàgon

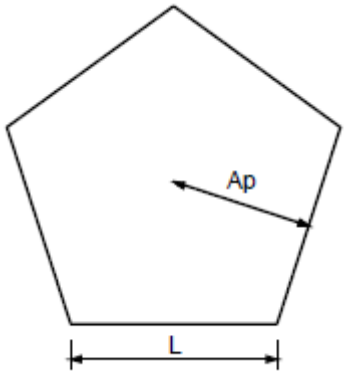

Figura 8.2.**Representació de les mesures realitzades en el pentàgon.**

$$Àrea\ pentàgon = \frac{5 \times L \times Ap}{2} \qquad \text{(Eq. 8.2)}$$

B	L	Ap	Àrea
34,4	22	15,383969	846,1182974
36,5	22	16,3231067	897,7708679
34,8	22	15,5628524	855,9568823
35	22	15,6522941	860,8761747
34,5	23	15,4286899	887,1496683
Mitjana			860,8761747

Graella 2Mesures pentagon

8.1.3. Quadrilàters

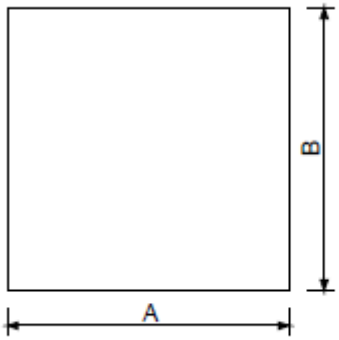

Figura 8.3.**Representació de les mesures realitzades en els quadrangulars.**

$$\grave{A}rea = A \times B \qquad \text{(Eq. 8.3)}$$

Quadrat petit

A	B	Àrea
25,5	26	663
25,2	26,8	675,36
Mitjana		669,18

Rectangle mitja

A	B	Àrea	Rel.costats
45,3	25,5	1155,15	0,562913907
44,9	25,6	1149,44	0,570155902
Mitjana		1152,295	0,566534905

Quadrat mitja

A	B	Àrea
37,7	36,6	1379,82
38	38,5	1463
Mitjana		1421,41

Rectangle gran

A	B	Àrea	Rel.costats
54,9	38	2086,2	0,692167577
54,8	37,8	2071,44	0,689781022
Mitjana		2078,82	0,6909743

Quadrat gran

A	B	Àrea
42	40,6	1705,2
42,3	41,5	1755,45
Mitjana		1730,325

Rombe petit

A	B	Àrea
30	25,3	759
31	24,6	762,6
Mitjana		760,8

Rectangle petit

A	B	Àrea	Rel.costats
36,8	24,5	901,6	0,66576087
35,7	24,7	881,79	0,691876751
Mitjana		891,695	0,67881881

Rombe gran

A	B	Àrea
32,6	30,2	984,52
34,7	31	1075,7
Mitjana		1030,11

Graella 3Mesures dels quadrangulars

8.1.4. Triangles

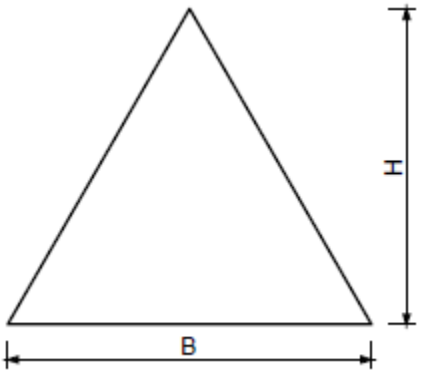

Figura 8.4.**Representació de les mesures realitzades en els triangles.**

$$\grave{A}rea = \frac{H \times B}{2} \qquad \text{(Eq. 8.4)}$$

Triangle equilàter

H	B	Àrea
26,3	29	381,35
24,7	31,6	390,26
25,85	29	374,825
Mitjana		381,35

Triangle isòsceles gran

H	B	Àrea
38,2	37,1	708,61
37,1	38,2	708,61
26,9	52,15	701,4175
Mitjana		708,61

Triangle isòsceles petit

H	B	Àrea
28	27,6	386,4
27,6	28	386,4
19,8	39	386,1
Mitjana		386,4

Triangle rectangle

H	B	Àrea
22	51,7	568,7
51,7	22	568,7
20,4	55,7	568,14
Mitjana		568,7

Graella 4Mesures triangle

8.2. **Ground truth**

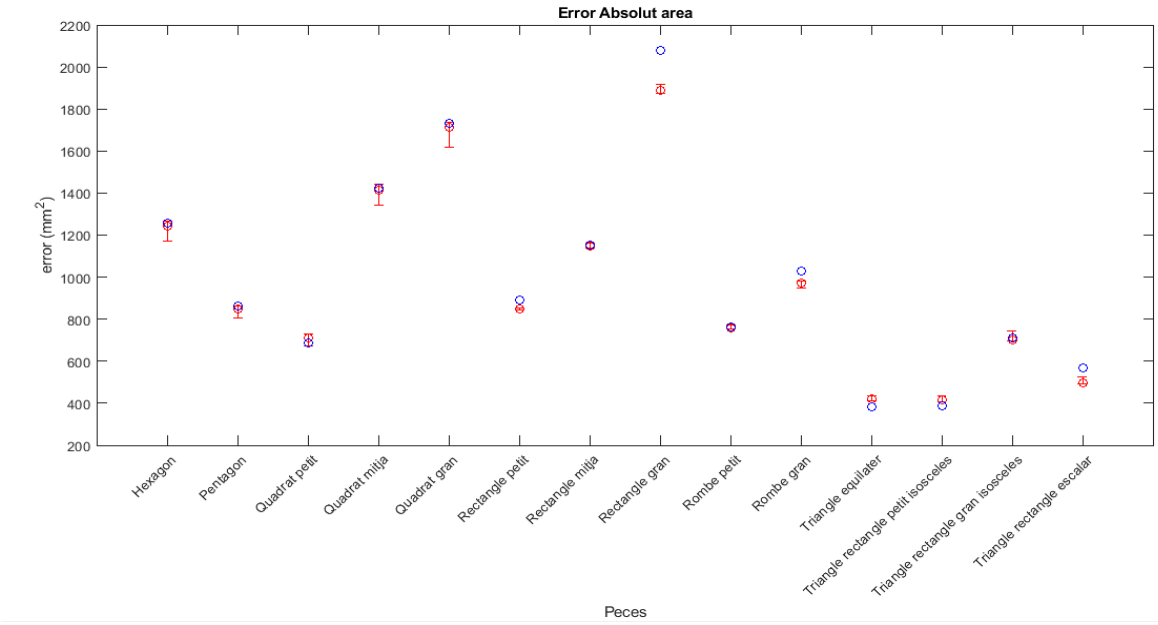

Figura 8.5.**Error absolut en el càlcul d'àrea segons la figura.**

El cercles blaus indiquen l'àrea calculada manualment i l'interval vermell indica el rang d'àrees obtingudes per cada peça.

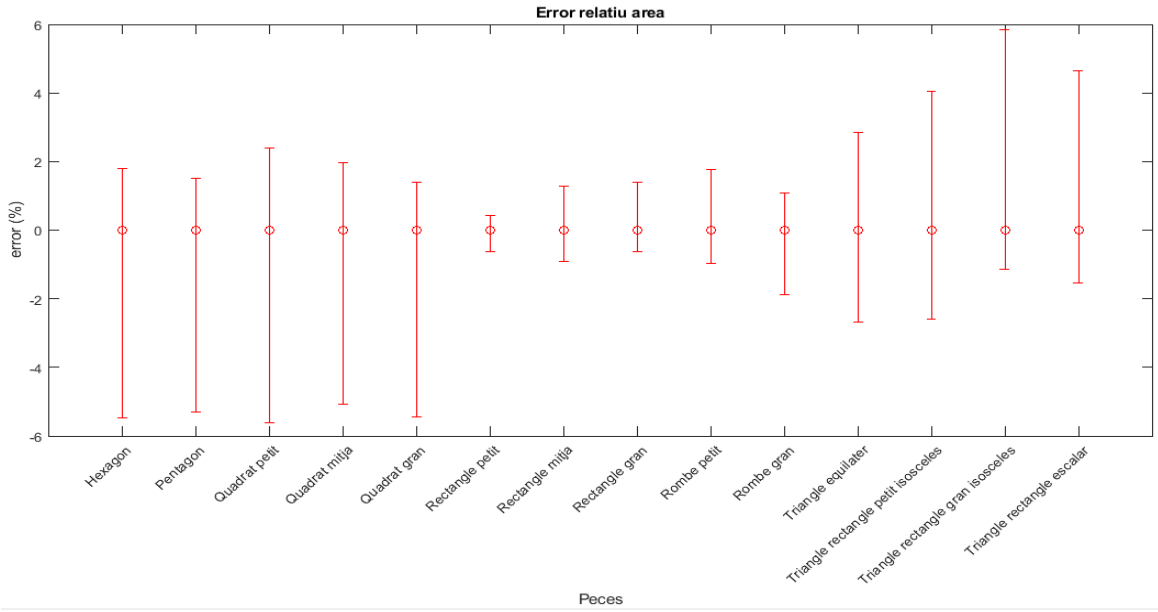

Figura 8.6.**Error relatius en el càlcul d'àrea segons la figura.**

L'interval vermell indica la diferència relativa entre la mitja i les mostres més dispars.

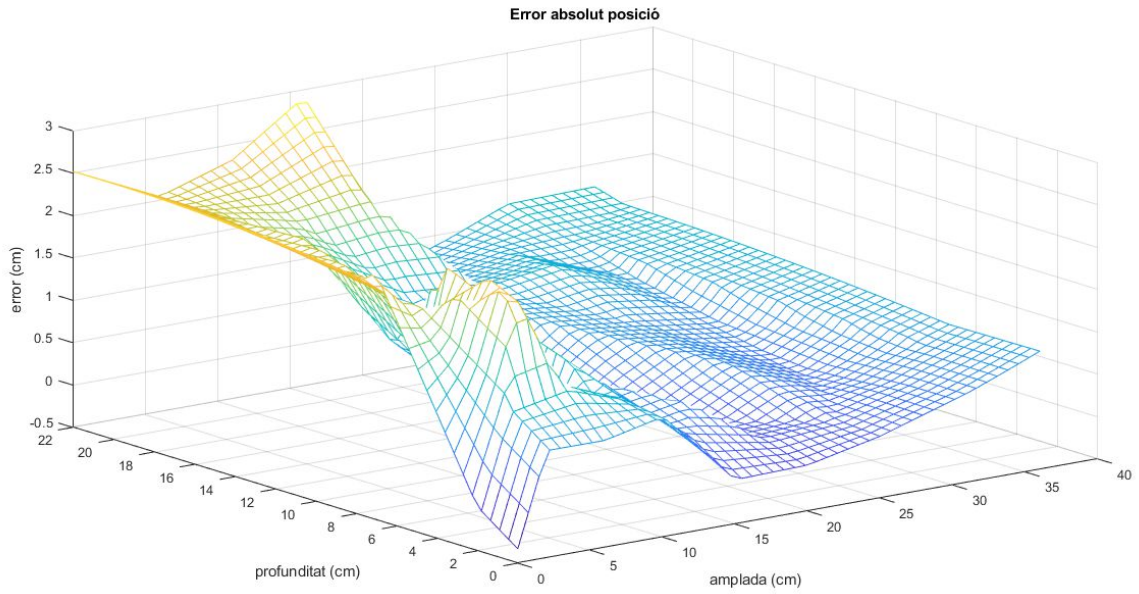

Figura 8.7.**Error absolut de l'àrea segons el posicionament, eixos X i Y representen la posició de la peça en el taulell (cm), eix Z indica la desviació de l'àrea real en mm^2.**

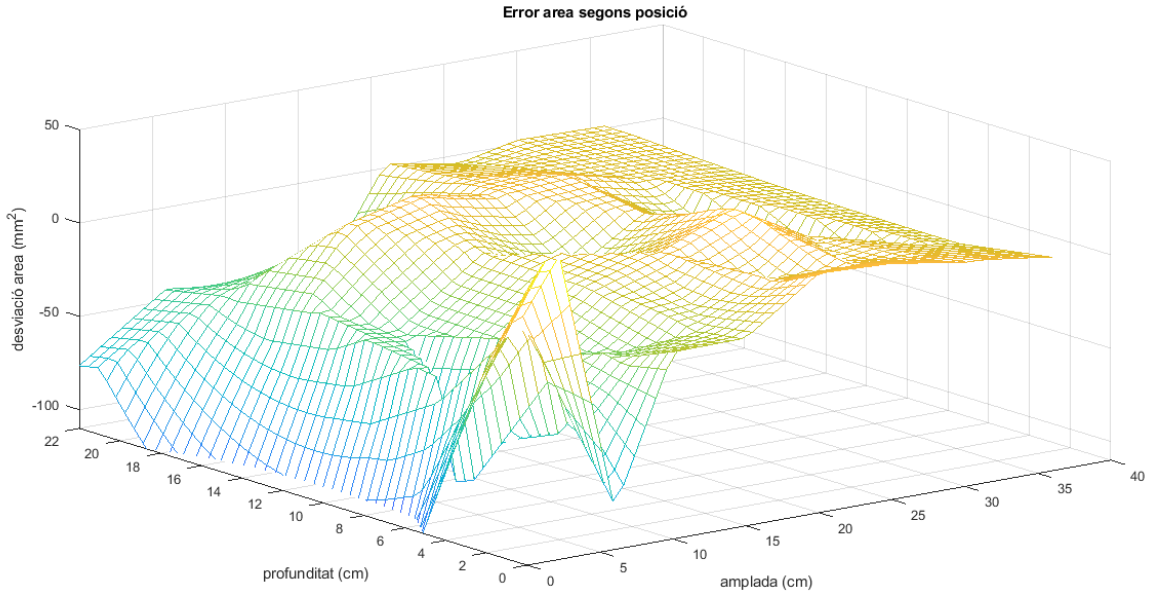

Figura 8.8.**Error absolut en el posicionament, eixos X i Y representen la posició de la peça en el taulell (cm), eix Z indica la desviació lineal (cm).**

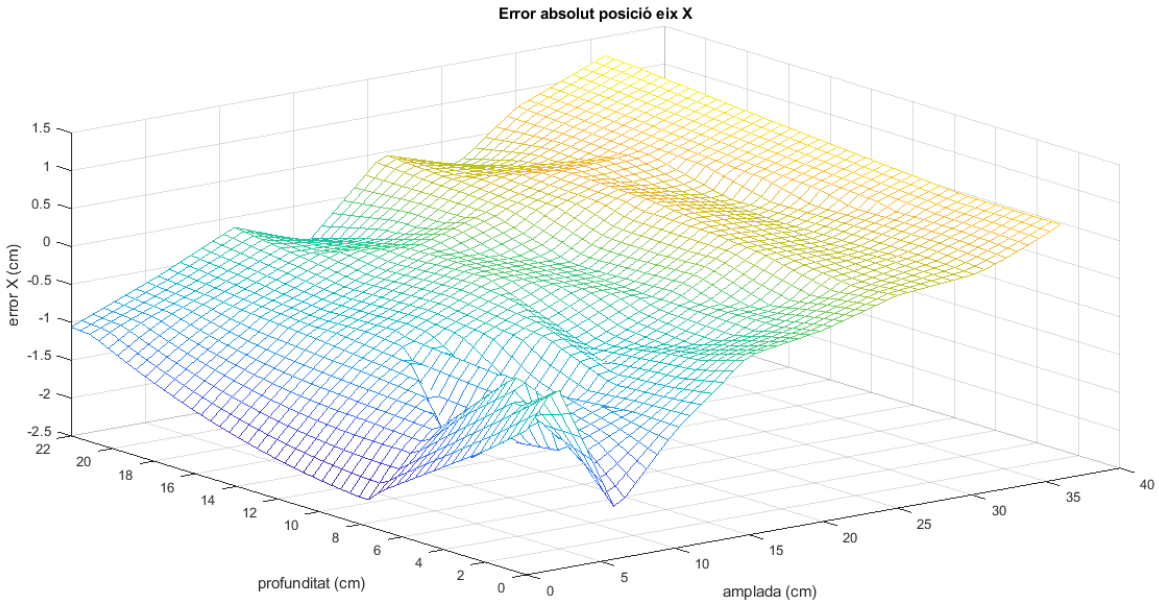

Figura 8.9.**Error absolut en el posicionament de l'eix X, eixos X i Y representen la posició de la peça en el taulell (cm), eix Z indica la desviació lineal (cm).**

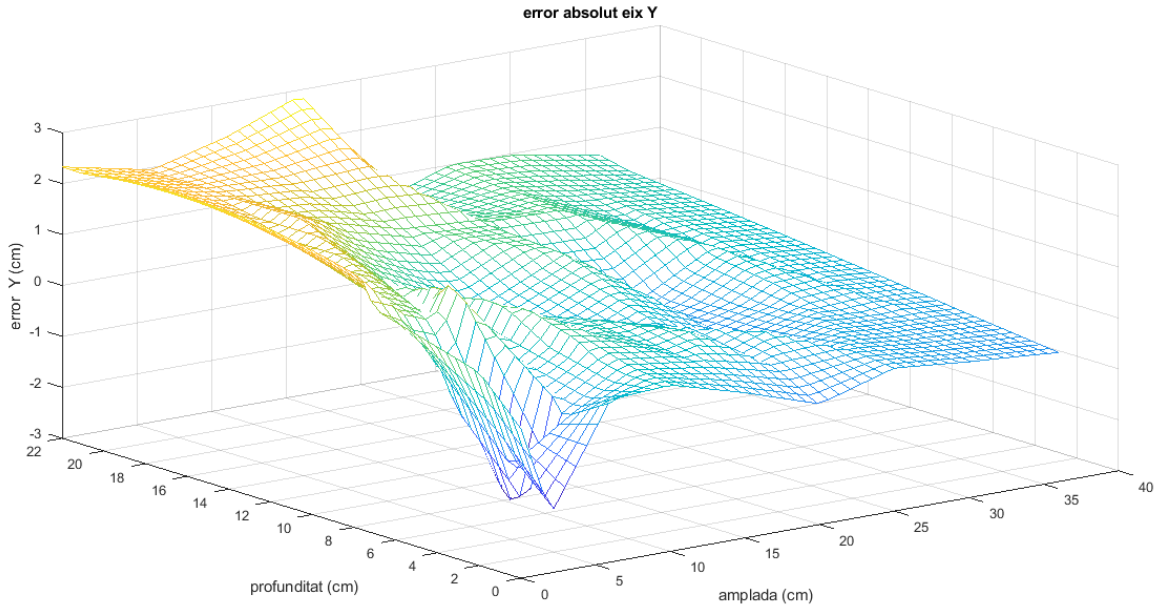

Figura 8.10.**Error absolut en el posicionament de l'eix Y, eixos X i Y representen la posició de la peça en el taulell (cm), eix Z indica la desviació lineal (cm).**

8.3. **Interpretació dels resultats**

La Figura 8.5 i la Figura 8.6 han sigut realitzades amb una població de 5 mostres, però són dades suficients per observar una disparitat.

Amb la Figura 8.5 es pot observar que, com s'explicava en la introducció, una classificació directe segons l'àrea no és possible ja que alguns comparteixen rang d'àrea, com per exemple les formes 2 i 6 que corresponen a pentàgon i rectangle petit. Per diferenciar entre aquestes dues peces només caldria la utilització del número de vèrtex, però també hi ha el cas de les formes 11 i 12 que corresponen a triangle equilàter i triangle rectangle isòsceles petit, on el número de vèrtex no permetria diferenciar-les.

També cal d'estacar una clara diferència entre els valors calculats manualment i els obtinguts amb el programa. Clar exemple és el del rectangle gran on les mostres quedaven lluny del valor esperat, és possible que això sigui degut a la forma irregular de les peces. Tot i així al no compartir rang d'àrees possibles amb la resta de rectangles no produeix cap error.

Amb la Figura 8.6 es pot observar que en el pitjors dels casos la diferencia relativa entre l'àrea major i la menor és d'un 8%.

Les figures Figura 8.7, Figura 8.8, Figura 8.9 i Figura 8.10 s'han realitzat amb una mostra de 40 posicions diferents repartides per tot el taulell. Com s'indicava a l'inici de l'apartat 8, la càmera ha estat col·locada a 55 cm del centre del taulell, 20˚ respecte la frontal i 64˚ respecte la vertical. És important destacar això ja que permet entendre les formes que es generen. També cal aclarir que els valors en els marges més allunyats són projeccions segons el que s'espera veient com estava en punts a 2 cm i que per tant no han de ser utilitzats com a referència.

Per finalitzar amb l'estudi de les àrees la Figura 8.8 mostra cóm l'àrea del quadrat petit varia segons el seu posicionament. En la major part es pot considerar una variació nul·la, però com més s'allunya del vèrtex on estava situada la càmera la diferencia comença a ser més pronunciada. Tot i així en la posició més allunyada segueix quedant dintre del rang que la determina.

Pel que fa a les figures Figura 8.7, Figura 8.9 i Figura 8.10 ,si no ens fixem en valors puntuals que queden dispars, es pot observar una clara desviació que com es comentava en el paràgraf anterior s'incrementa segons la posició s'allunya del vèrtex on està situada la càmera. Tot i així, es pot fer una clara diferencia entre els valors de Figura 8.9 i Figura 8.10, i és que uns augmenten i els altres disminueixen.

Conclusions

Un cop realitzades les proves de funcionament que es mostren en l'apartat anterior és el moment d'extreure conclusions. Les proves s'han realitzat amb la càmera en una posició no òptima per tal de mostrar un cas real (a 55 cm del centre, 20˚ respecte la frontal i 64˚respecte la vertical)

1. El sistema de reconeixement de figures no mostra cap problema per identificar-les independentment de la posició o orientació.
2. Tot i utilitzar colors vius per millorar la segmentació, segons com sigui la il·luminació, encara es produeixen errors ocasionalment.
3. El reconeixement de figures és gràcies a l'èxit en la transformació de la imatge.
4. El procés és un pel lent.
5. La nota discordant és el posicionament, una desviació tant gran fa que actualment no sigui útil per superfícies d'aquesta mida. És possible que amb alguna de les millores futures que s'exposen a continuació aquesta desviació es corregeixi i pugui ser utilitzat perfectament.

Degut a que el posicionament és irregular resulta necessari desenvolupar alguns punts que poden ser causants d'aquest problema.

Arreglar la correcció d'inclinació. La correcció d'inclinació és la encarregada d'alienar amb l'horitzó la càmera. El sistema actual és lent i rudimentari. No té en compte com afecta a les posicions elevades ni a la distorsió per la profunditat. Pot ser interessant que, en comptes de rotar la imatge, es treballi amb una línea d'horitzó inclinada.

Una altre solució al problema de la desviació és treballar amb més TAGs sobre la superfície, creant així petites regions, i que aquestes ocupin tota la zona de treball del robot.

A sobre, també hi ha problemes en posicions on 3 TAGs queden al mateix canto de l'eix de X_Fuga. S'haurien de crear les excepcions corresponents en el codi per a que actués de forma específica en cas de trobar-ho.

Pressupost i/o Anàlisi Econòmica

En aquest capítol es realitzarà una estimació del cost de la realització del present treball.

8.4. Hardware

Concepte	Preu unitari	Quantitat	Subtotal
Càmera web logitech c 270	24,95 €	1	24,95 €
Ordinador Intel i5-3570 3,4 Ghz	713,85 €	1	713,85 €
Subtotal			**738,80 €**

8.5. Software

Concepte	Preu unitari	Quantitat	Subtotal
Windows 10 Profesional	23,90 €	1	23,90 €
MATLAB Student	35,00 €	1	35,00 €
MATLAB Computer Vision System Toolbox	7,00 €	1	7,00 €
MATLABAcquisitionToolbox	7,00 €	1	7,00 €
MATLABProcessingToolbox	7,00 €	1	7,00 €
MATLAB Coder	7,00 €	1	7,00 €
MATLABParallel Computing Toolbox	7,00 €	1	7,00 €
Subtotal			**93,90 €**

8.6. Miscel·lània

Concepte	Preu unitari	Quantiat	Subtotal
Figures	1,00 €	14	14,00 €
Suport càmera	16,99 €	1	16,99 €
Plataforma	24,00 €	1	24,00 €
Plantilla base	0,90 €	4	3,60 €
Plantilla calibratge	0,08 €	1	0,08 €
Subtotal			**58,67 €**

8.7. Ma d'obra

Concepte	Preu unitari	Quantitat	Subtotal
Enginyeria	40,00 €	600	24.000,00 €
Redacció	35,00 €	40	1.400,00 €
Construcció	35,00 €	20	700,00 €
Subtotal			**26.100,00 €**

8.8. Cost total

Concepte	Subtotal
Hardware	738,80 €
Software	93,90 €
Miscel·lània	58,67 €
Ma d'obra	26.100,00 €
Total	**26.991.37 €**

Bibliografia

[1] Terven Salinas, J., Salas, J. andRaducanu, B. (2013). Estado del Arte en Sistemas de Visión Artificial para PersonasInvidentes. KomputerSapiens, I(V), pp.20-23.

[2] Betancor Pérez, A. (2008). Sistema De Reconocimiento de MatrículasBasado en Visión Artificial para Control de Acceso. UNIVERSIDAD POLITÉCNICA DE CARTAGENA.

[3] Delgado Hernández, M. (2016). Visión artificial aplicada a la robótica. La Laguna: Universidad de La Laguna. Availableat: https://riull.ull.es/xmlui/bitstream/handle/915/2914/Vision%20artificial%20aplicada%20a%20la%20robotica.pdf?sequence=1 [Accessed 20 Apr. 2018].

[4] Mery, D. (2018). Vision por computador. Santiago de Chile: UniversidadCatolica de Chile. Availableat: http://dmery.sitios.ing.uc.cl/Prints/Books/2004-ApuntesVision.pdf [Accessed 20 Apr. 2018].

[5] PERSPECTIVA CÓNICA. (2018). Sevilla: I.E.S. TORRE DE LOS GUZMANES, p.10. Availableat: http://www.areadedibujo.es/documentos/2-bachillerato/conica/perspectiva-conica.pdf [Accessed 20 Aug. 2017].

[6] Bouguet, J. (2015). CameraCalibrationToolbox for MATLAB. [online] http://www.vision.caltech.edu. Availableat: http://www.vision.caltech.edu/bouguetj/calib_doc/ [Accessed 22 Apr. 2018].

[7] En.wikipedia.org. (2018). Cardinal point (optics). [online] Availableat: https://en.wikipedia.org/wiki/Cardinal_point_(optics)#Principal_planes_and_points [Accessed 22 Apr. 2018].

[8] Weng, J., Cohen, P. andHerniou, M. (1992). TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. 10th ed. Auckland, p.968.

[9] Physics.stackexchange.com. (2016). What is the "tangential" distortion of OpenCVactuallytangential to?. [online] Availableat: https://physics.stackexchange.com/questions/273464/what-is-the-tangential-distortion-of-opencv-actually-tangential-to [Accessed 22 Apr. 2018